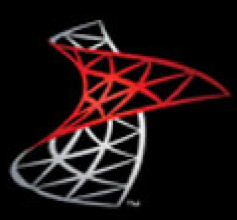
برمجة قواعد البيانات باستخدام

Microsoft

SQL Server 2008



Microsoft

Visual C# 2010



المرجع العربى الأول

حسام كمال محمد

بسم الله الرحمن الرحيم

مقدمة

الحمد لله كما يُحب ربنا ويرضى ، والصلاة والسلام على عبده المجتبى ، سيدنا محمد ﷺ ،ورضي الله عن صحابته الأطهار وزوجاته امهات المؤمنين وبعد..

فهذا الكتاب الذي بين يديك هو عمل منقوص بلاشك من جهتين أن الانسان لا يصل للكمال بطبيعة حاله ، وأنه ليس العمل النهائي الذي أنشده ، تعجلت في طرحه على الرغم من توقعي أنه لم يبلغ ثلث ما أعد له ، ولكن حتي يتواصل مع جمهور المهتمين بإثراء المحتوى العربي من كتب البرمجة لكي أُحسن هذا العمل وغيره من الأعمال المستقبلية إن شاء الله ، الكتاب قريب من 100 صفحة بدأت فيه بالمعلومات الضرورية عن عالم قواعد البيانات بشكل موجز واكاديمي ، يسهل على أي مبتدئ فهمه وإن تعثر فقد تركت مفاتيح لكي يقوم بالبحث بها بنفسه لأن عالم البرمجة لاتتوقع أن يطعمك كل انسان كافة احتياجاتك للعلم ، كما وأن هذا الكتاب ليس لتعليمك لغة C# يُمكنك ايجاد عشرات المراجع للبداية في تعلم هذه اللغة الرائعة ، وسأفترض ان لديك الحد الأدني من المعرفة بها.

وأما عن تقسيمة الكتاب فهو مقدمة لكيفية تثبيت **SQL Server 2008** وكيف تحصل على المنتج بصورة مجانية لكن بنسخة أقل من حيث الامكانيات وهي النسخة **Express** ، لطمأنتكم فالأكواد الموجودة في الكتاب تم تطبيقها على كل نسخ **SQL Server** ابتداءا من **2005** إلى **2008** وحتى النسخة **SQL Server 2008 R2** بكافة اصداراتهم بدون أي مشاكل ، وستلاحظ هذا من خلال الصور الملتقطة للعمل على **SQL Server** بدون التنويه على تغير النسخة لأنها ستعمل معك بكفاءة مع أية نسخة،ولو صادفك أية مشكلة يمكنك التواصل معي لمساعدتك بقدر الإمكان ، ثم يستمر الكتاب باستعراض مبادئ العمل في بيئة **SQL Server** والتعرف على كيفية التعامل مع البيانات من ادخال وحذف وتعديل ، ثم يأتي دور التعرف على **Stored Procedures** من كيفية انشاءها والعمل عليها ، وتضمينها داخل أكواد **C#** وبالمناسبة أيضاً الأكواد تعمل على بيئة الدوت نت **2.0** إلى **4.0** بلا أي مشاكل أيضاً راعيت هذا اثناء الإعداد للكتاب ،ثم نتعرف على **XML** وكيفية تعاملها مع البيانات وفهم خصائصها ، ثم نختم بالتعامل مع العمليات **Transactions** وكيفية كتابة كود **ADO.NET** لهذه العمليات.

ما سوف أمضي فيه إن شاء الله في تطوير الكتاب واصدار مزيد من الفصول سوف يتعمق في **ADO.NET** والتعرف على مبادئ كتابة برامج قواعد بيانات لتطبيقات الويندوز وكذلك الويب.

أرجوا أن تستمتعوا بالكتاب والرجوع إلي حال مصادفتكم أيه مشاكل أثناء عملكم عليه.

في الختام أشكر كل من شجعني وثابر معي وسيثابر لأن العمل لم ينتهي بعد ،وأخص بالشكر زوجتي وأخوتي وأصدقائي فى العمل على حثهم لإنهاء هذا الكتاب جزاهم الله خيراً.

وفقنا الله وإياكم لما يحبه ويرضاه.

حسام كمال محمد
hosam@live.co.uk

نبذة عن المؤلف



حاصل على بكالوريوس الحاسبات والمعلومات ٢٠٠٨ – جامعة المنصورة –
وأعمل مطور شيربوينت بشركة Egypt Network ،لدي الخبرة وإجادة
العمل بكل من C#,ASP.NET,SQL Server,SharePoint ,InfoPath,XML
يمكنك متابعة مدونتي التقنية فيما يخص الشيربوينت وكذلك بيئة الدوت نت
والتواصل معي على الرابط التالي:
http://sharepointhero.wordpress.com/

الفهرست

رقم الصفحة

قبل الشروع في العمل

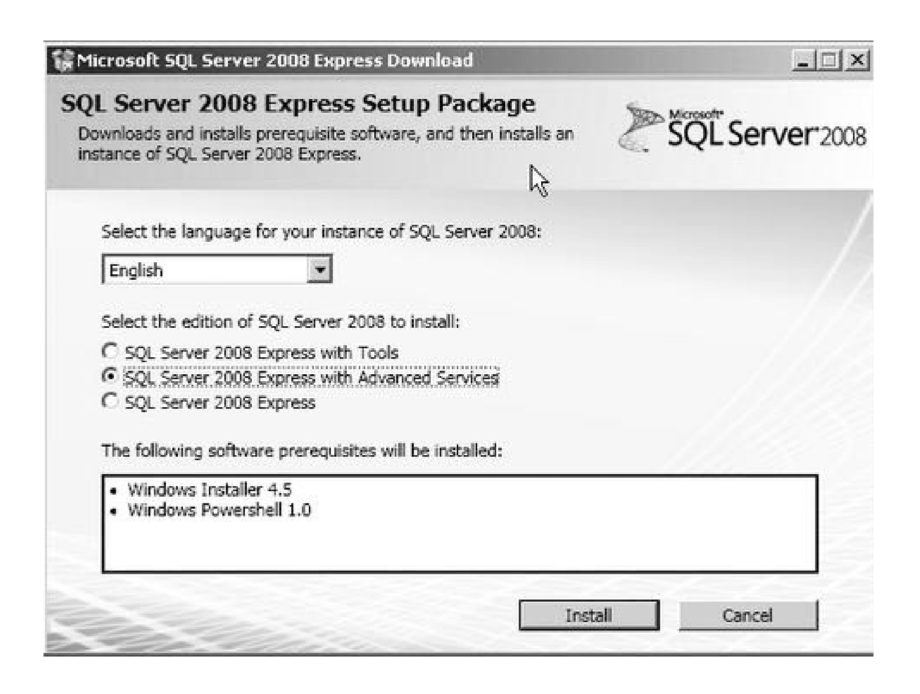
تثبيت MS SQL Server 2008

متطلبات ما قبل التثبيت

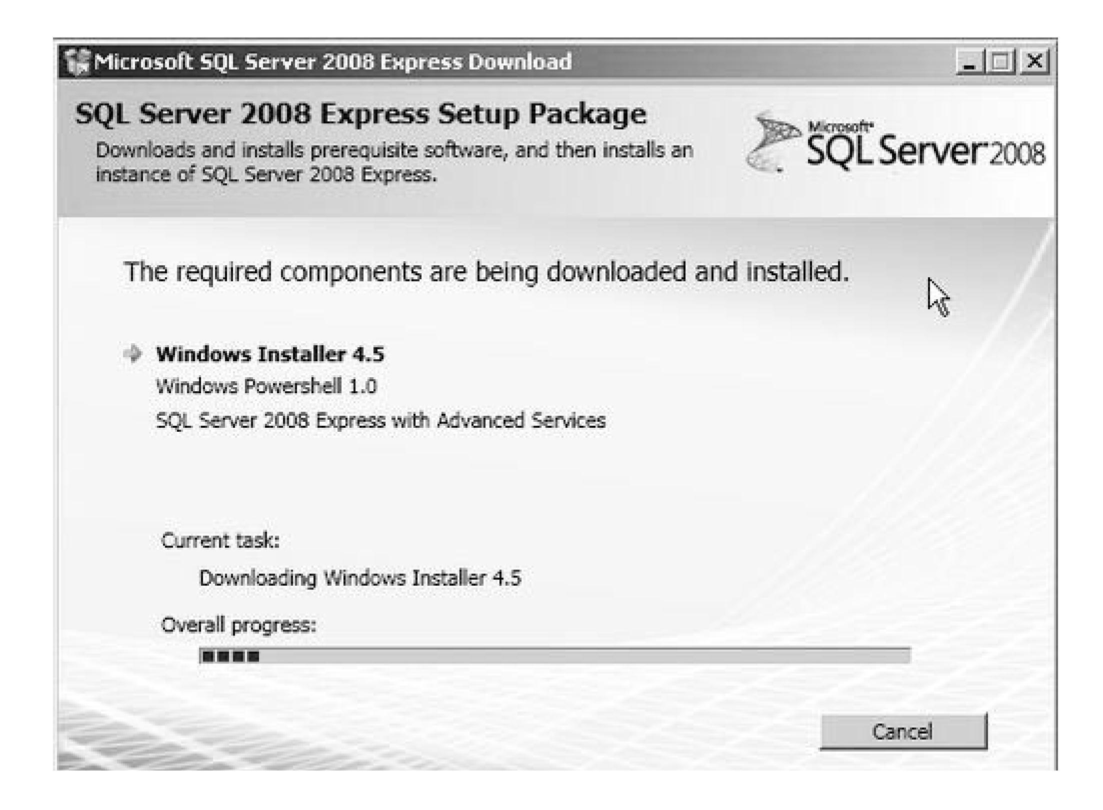
لابد من توفر الحد الأدني من الامكانيات التالية لكي تستطيع تثبيت MS SQL Server 2008 إلى جهازك ، ونعني بالجهاز هنا الكمبيوتر الشخصي أو حتى جهاز خادم:
مبدئياً سنحدد ما اذا كنا سنعمل على النسخة الخفيفة Express أم النسخة الكاملة ، والفارق بينهما كبير اختصاراً فهو يكمن في دعم عدد من الاجراءات والعمليات وكذا مساحة قاعدة البيانات وعدد دعم المستخدمين المتصلين في المرة الواحدة.
سنتحدث عن الحد الأدني وهو النسخة الخفيفة وفيما عدا ذلك قم بالبحث عن توافق النسخة الكاملة مع امكانياتك وكذا طريقة التثبيت التي لن تختلف عن النسخة Express التي نحن بصدد تثبيتها يُمكنك اذا استشعرت الملل من متابعة الخطوات مشاهدة هذا الفيديو على يوتيوب (لست مسئولاً عن تغير الرابط في أي وقت):
http://www.youtube.com/watch?v=CXJP4D503Tk

أو تابع الشرح معي...
أولاً :
لابد من توافر أحد هذه الأنظمة على جهازك :
Windows Server 2003 Service Pack 2, Windows Server 2008, Windows Vista, Windows Vista Service Pack 1, Windows XP Service Pack 2, Windows XP Service Pack 3 ,Windows 7.

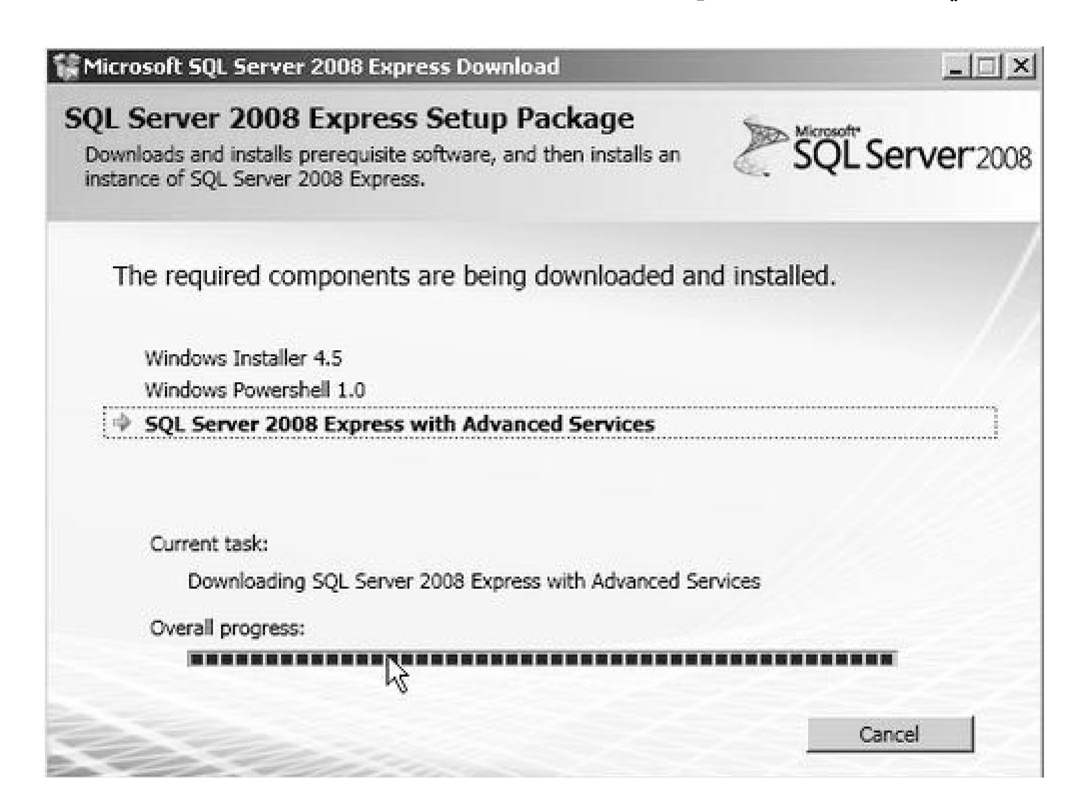
ثانياً :
الحد الأدنى من الذاكرة المتطلبة ٢٥٦ لكن يفضل ١ جيجا على الأقل وأعتقد أن اغلب الأجهزة تحوي هذا الطلب ، كذلك توفر مساحة على القرص الصلب لا تقل عن ٢ جيجا وكلما زادت كلما كان أفضل ومعالج قوته لا تقل عن ٢ جيجا هرتز سواءاً ٣٢ بت او ٦٤ بت ، ويفضل أن يكون من طراز انتل x ٦٤ بت لأن النسخة Express لا تعمل مع معالجات IA64 القديمة.
للحصول على نسخة Express يُمكنك متابعة الرابط التالي (مع التنويه أنه عرضة للتغير):
http://www.microsoft.com/en-us/download/details.aspx?id=1695
وستلاحظ وجود نسختين احدهما تسمى SQLEXPR_x64_ENU.exe وهي لمعالجات ٦٤ بت ، والأخرى تسمى SQLEXPR_x86_ENU.exe وهى لمعالجات ٣٢ بت التي تتوفر في اغلب الأجهزة الشخصية ، قم بتحميل ما يناسبك من كلتا النسختين ، وقبل الشروع في التثبيت تأكد من توفر بيئة الدوت نت 3.5 sp1 ويمكنك البحث عنها او تنزيل الاصدار الرابع فهذا أفضل (بديهي عملك على فيجول ستوديو ٢٠١٠ يتضمن تثبيت بيئة الدوت نت) وكذلك ينبغي توفر Windows Installer 4.5.
بعد التحميل قم بالتثبيت وسيظهر لك هذه الاختيارات ..
قم باختيار الخيار الأوسط كما بالشكل وهو احد الخيارات في عملية التركيب ، والتي يسمح لنا بمشاهده أكبر عدد ممكن من الخدمات:



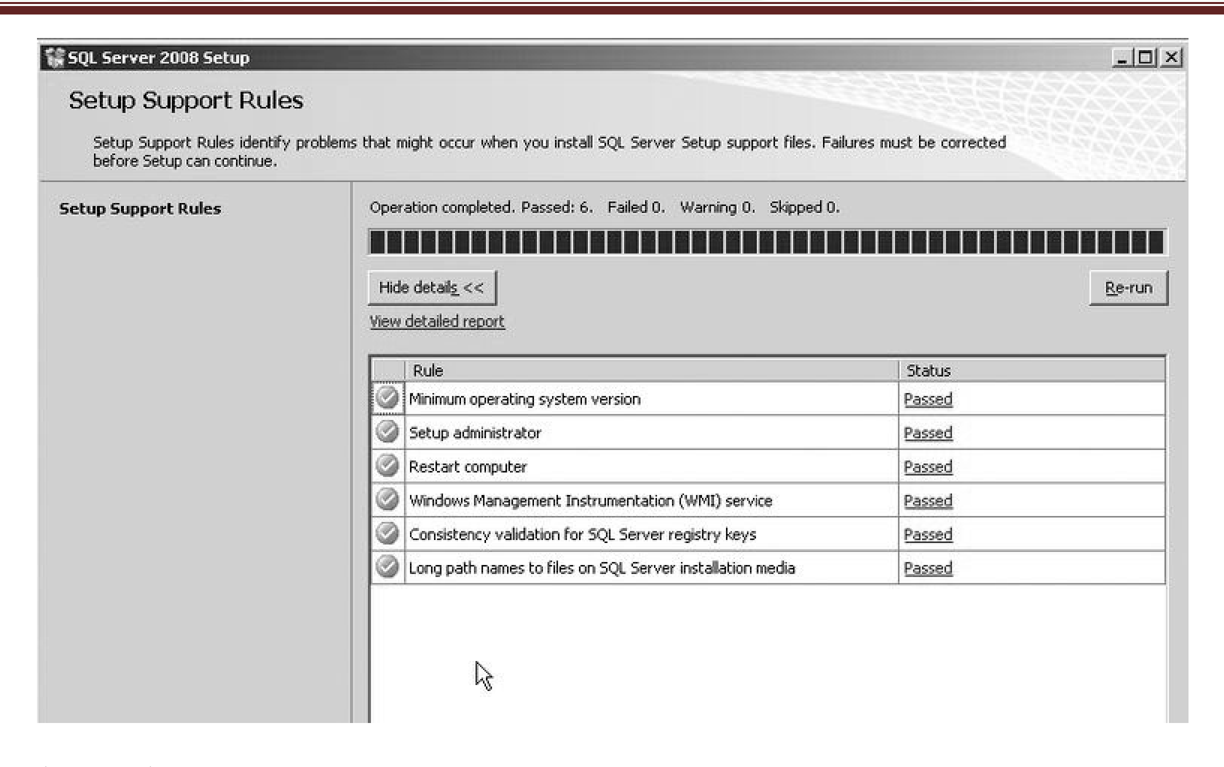
اختر Install ثم ترى النافذة التالية اذا كانت احدي هذه المتطلبات غير موجودة :



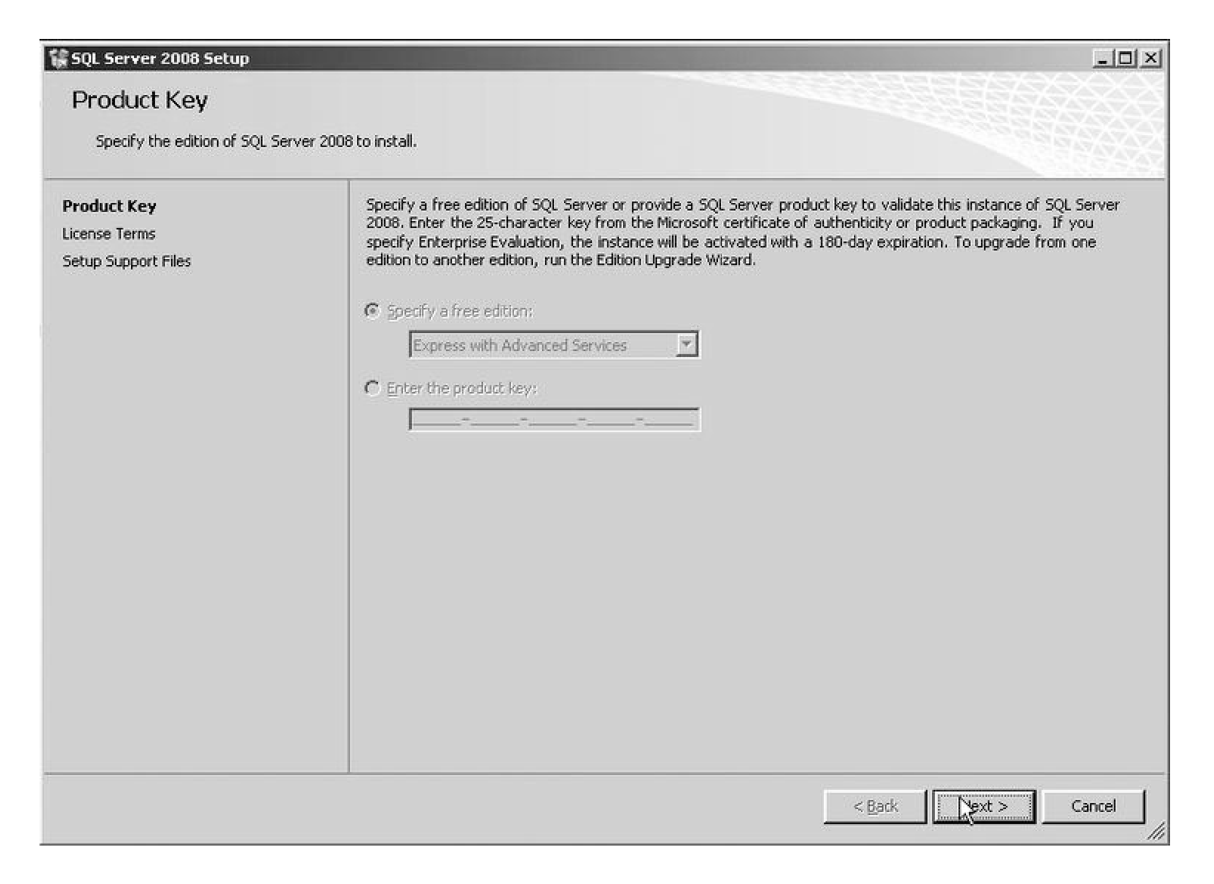
ثم يتتابع التثبيت ليبدء في تثبيت مكونات SQl Sevre Express:



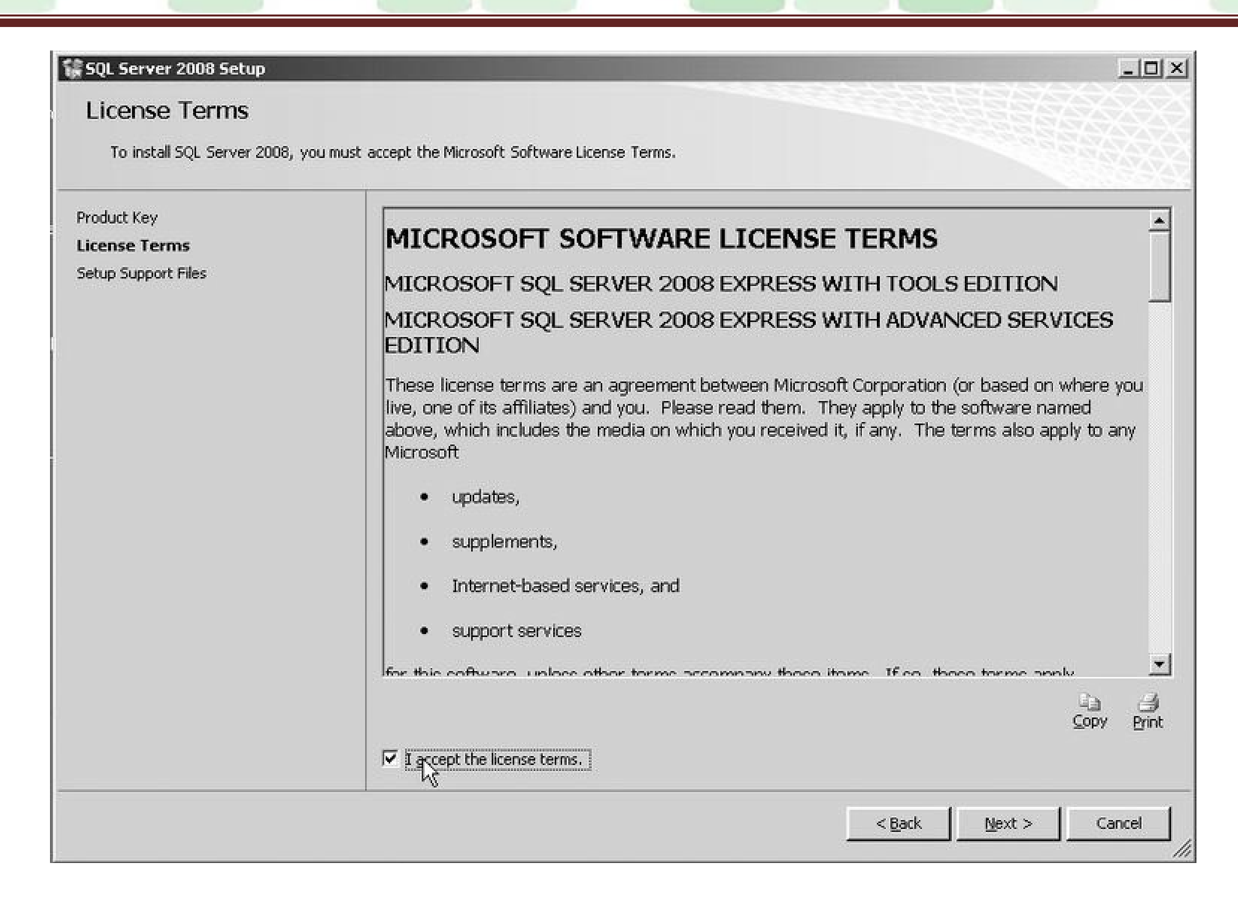
ثم يتبع بالتالي وهي قواعد التثبيت :



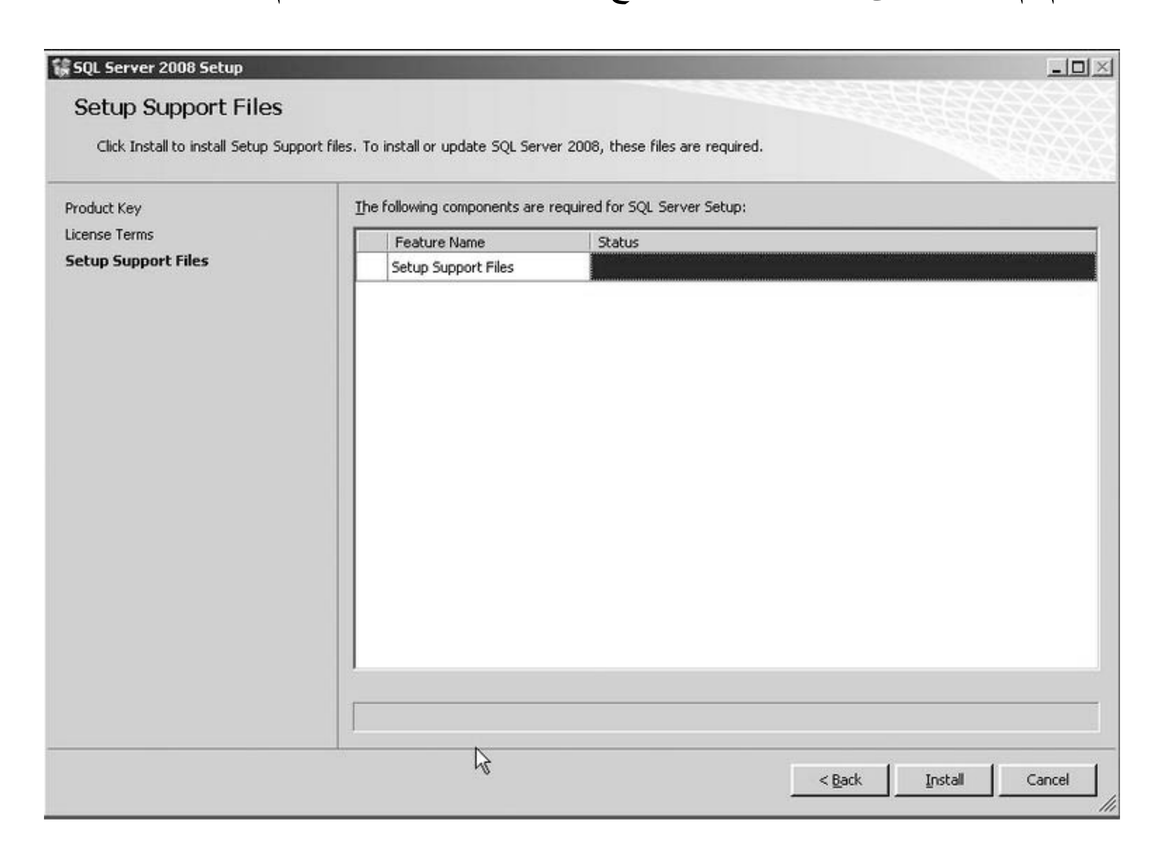
بعد الضغط على Next يتم الذهاب إلى شاشة رقم المنتج حيث يتم وضع الرقم للمنتج ..لكن هنا لا نحتاج أي رقم لأن البرنامج عبارة نسخة Express أنظر الشاكل التالي:



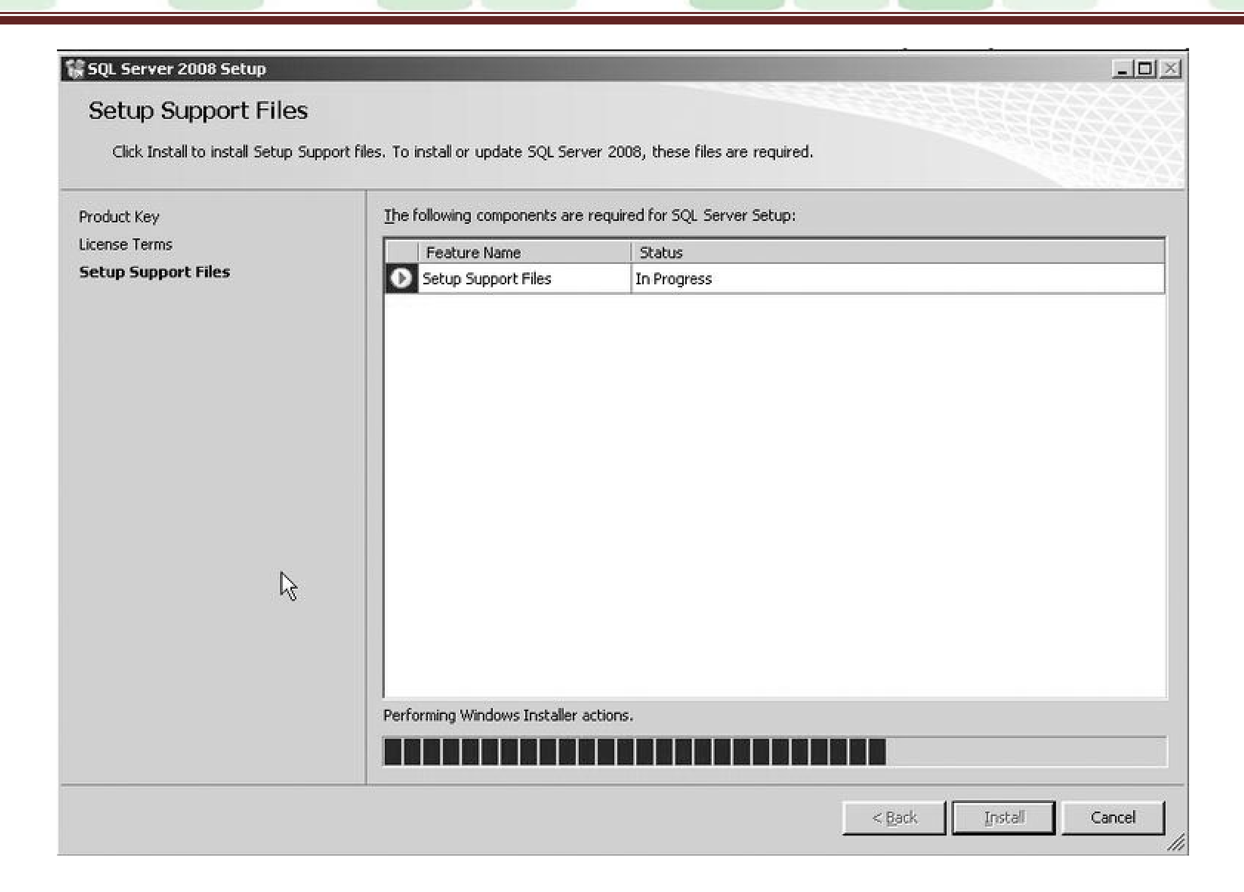
بعد هذه الخطوة يتم الذهاب غلى شاشة قبول الترخيص ،يتم إختاير الخيار I Accept the license Terms،بعد ذلك يتم الضغط على Next كما في الشكل التالي:



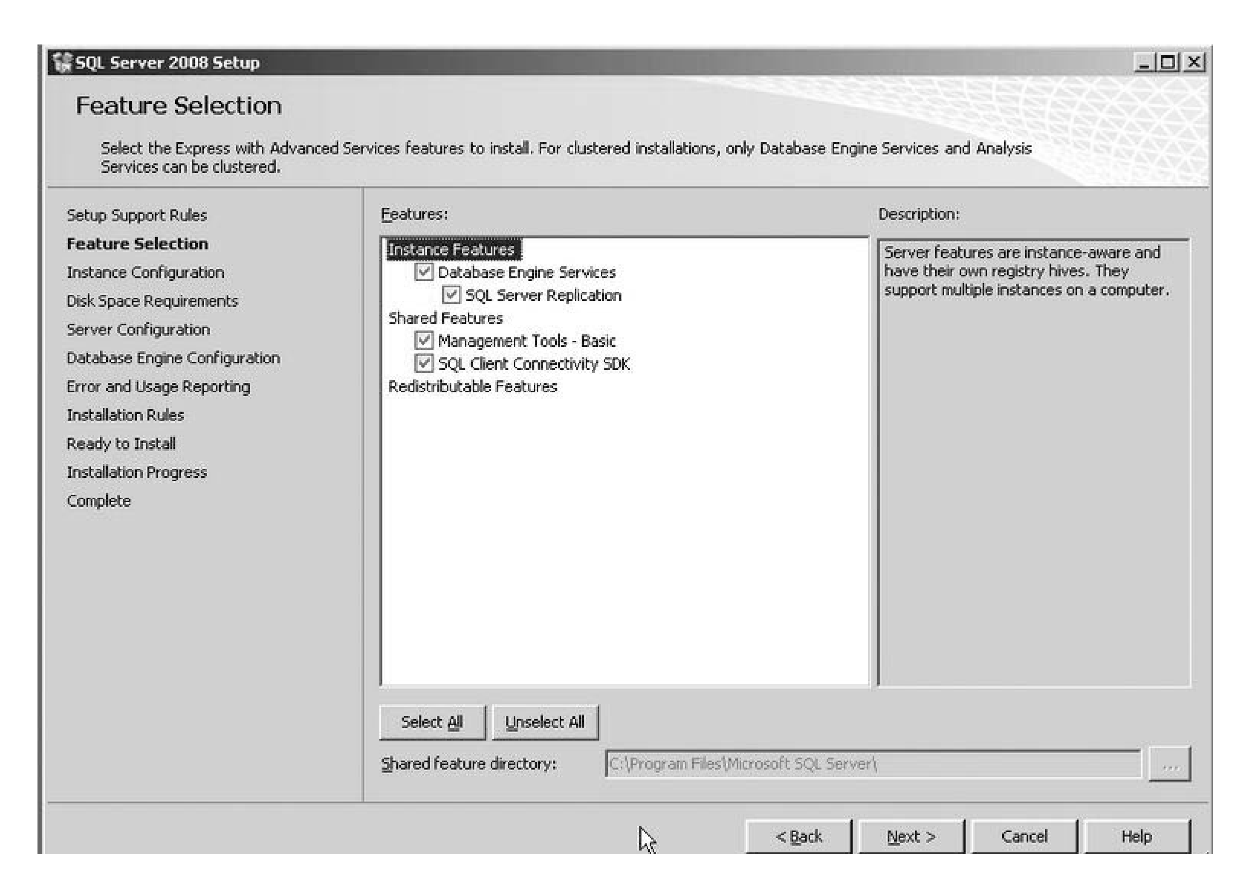
يتم أختيار ملفات الدعم ثم الضغط على Install ،فيبدء المعالج بإضافة ملفات أساسية للخادم كما بالشكل:



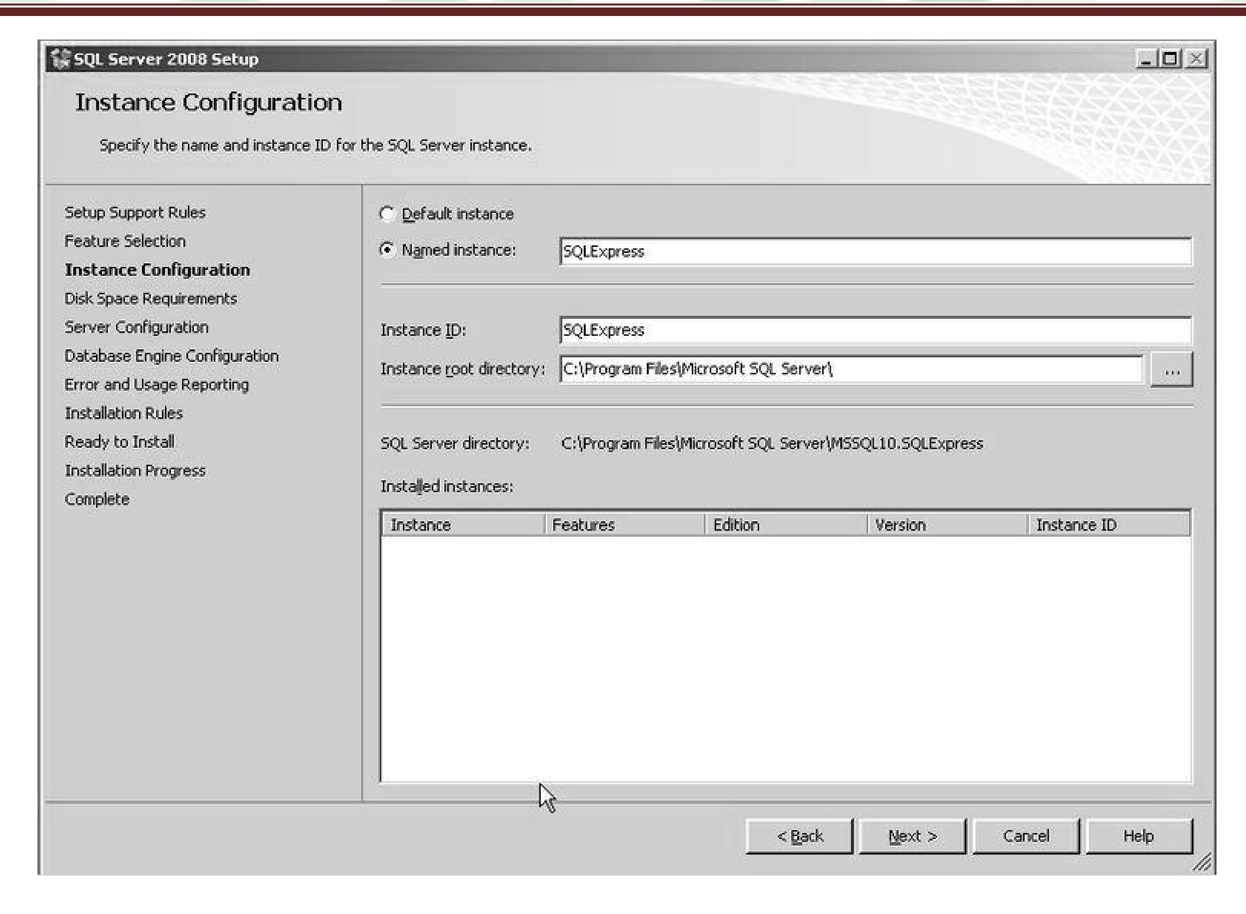
بعد ذلك يتم عمل فحص لقوانين التنصيب مرة أخرى كما في الشكل التالي:



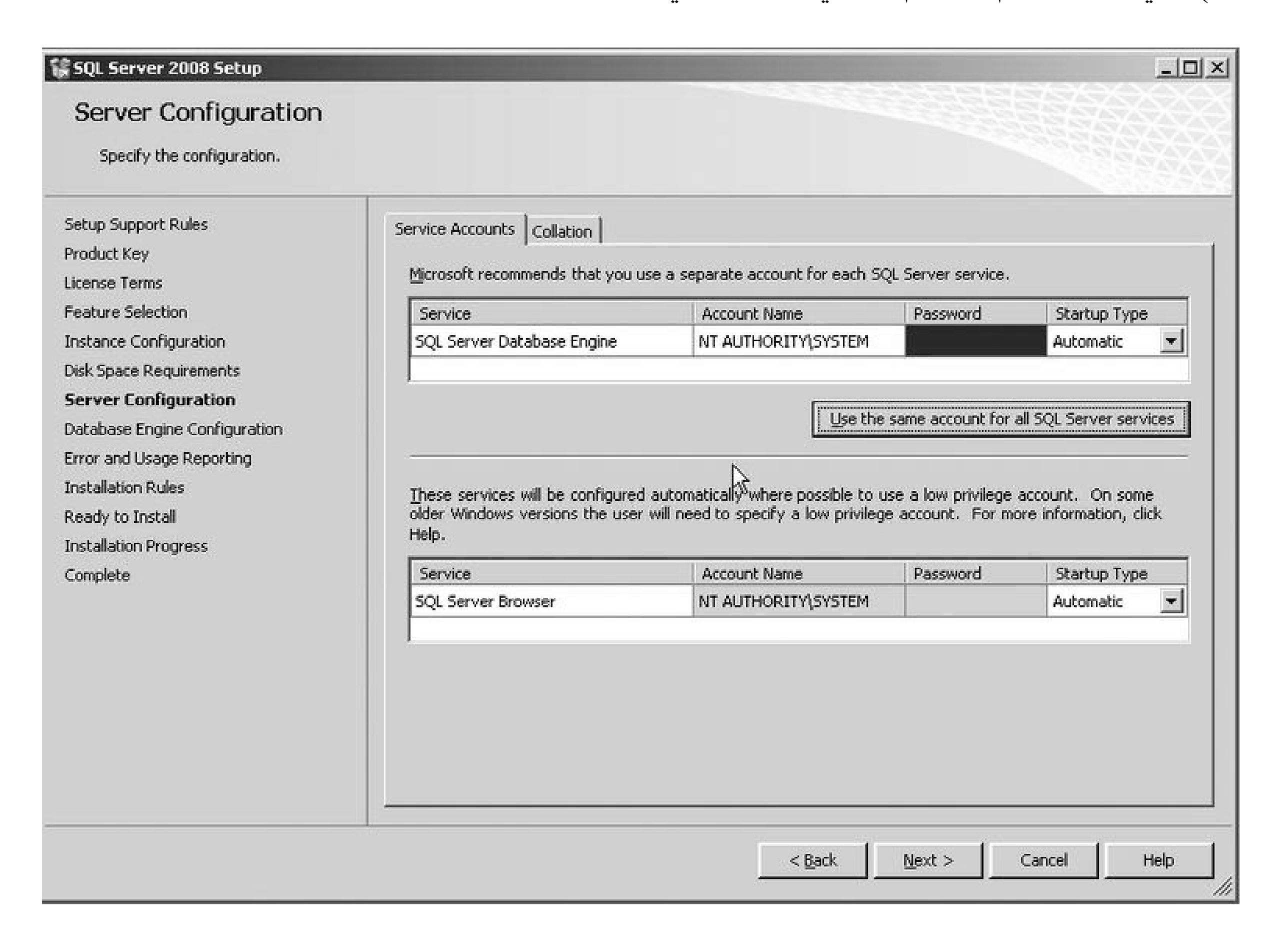
عند الضغط على Next نذهب إلى مرحلة خيارات الخواص للتركيب ، نختارهم كلهم كما في الشكل التالي:



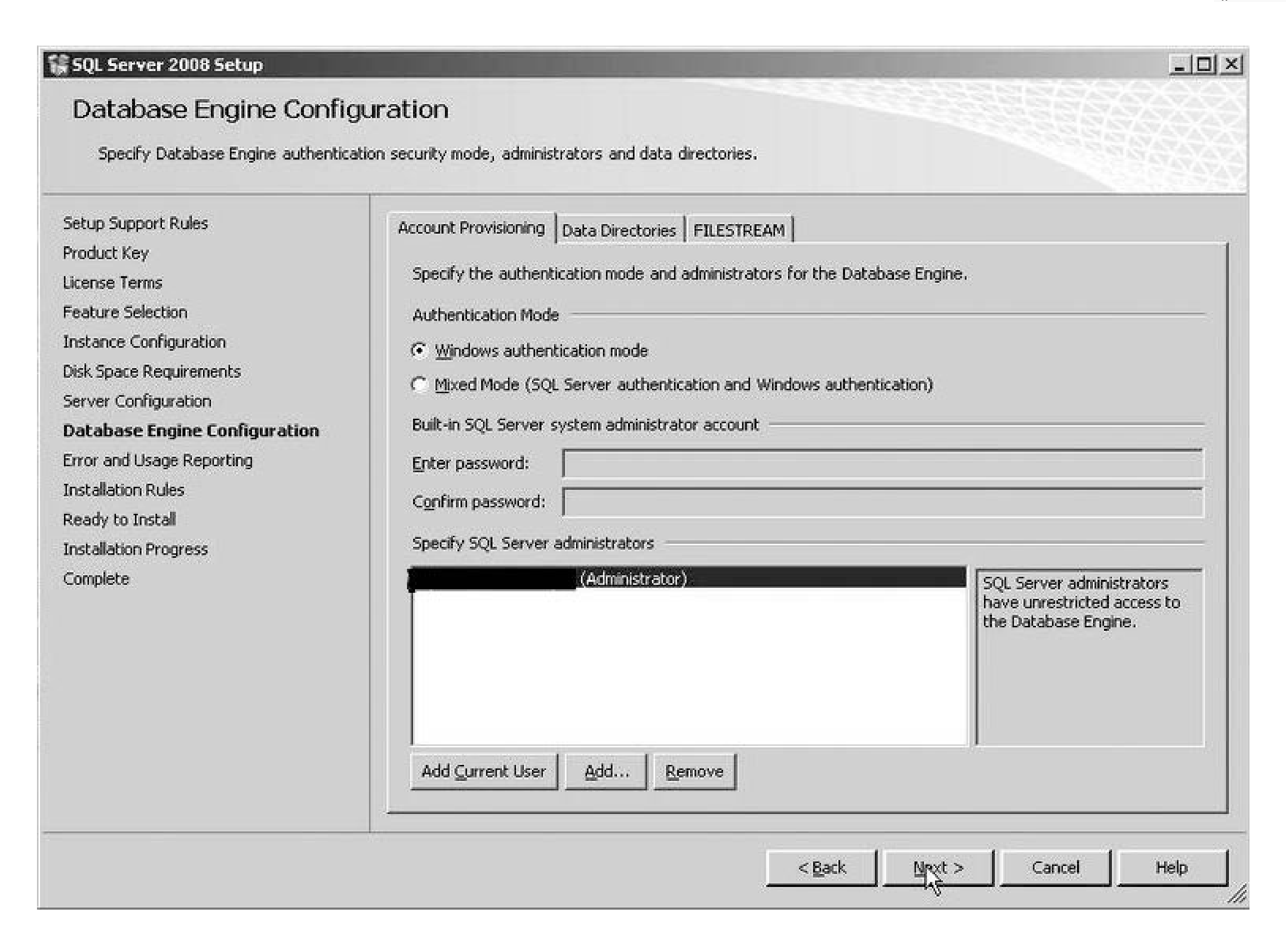
بعد الضغط على Next نذهب إلى شاشة اعدادات الـ Instance ،حيث يمكن أن نختار إسم لـ Instance . كما في الشكل التالي:



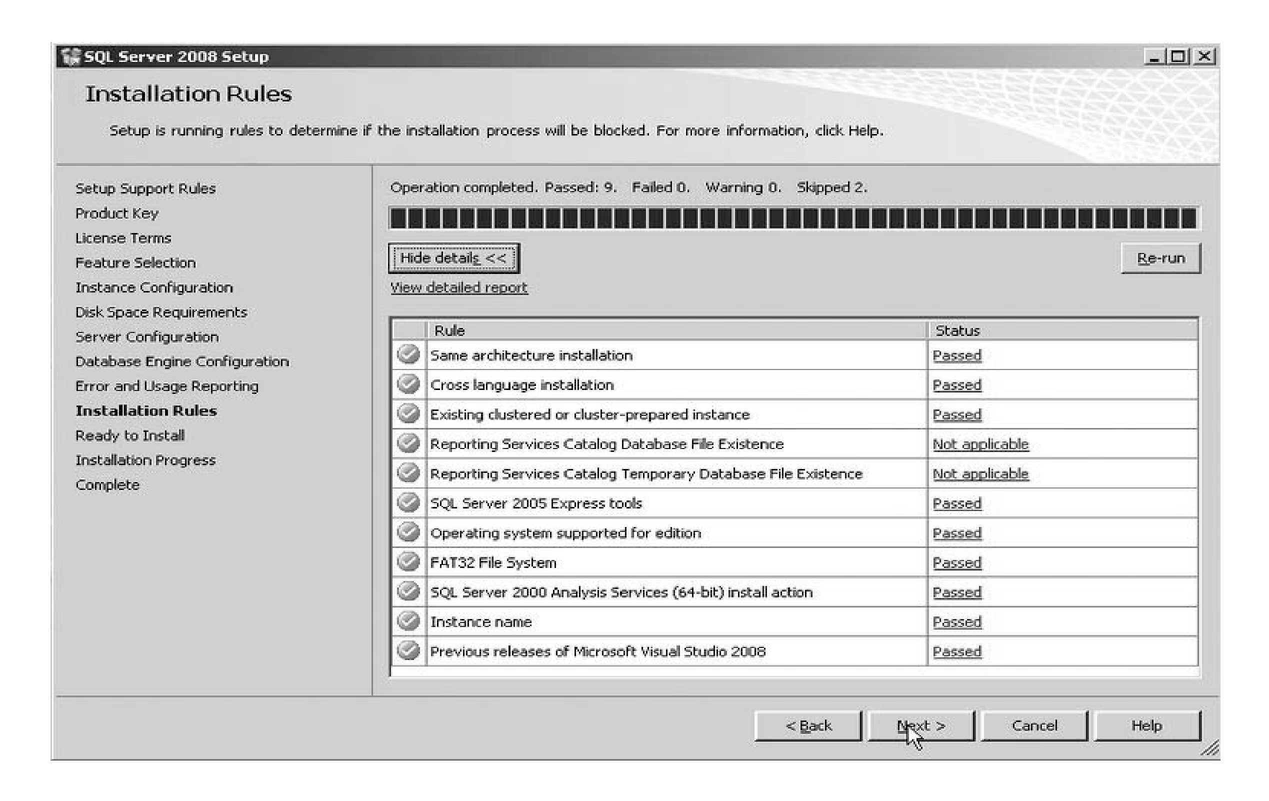
بعد الضغط على Next نذهب إلى شاشة تحديد الـ Service Accounts قم بادخال اسم الحساب الأدمن (اسم المستخدم خاصتك وكلمة المرور) وفي الغالب سيتم ادخالهم كما في الشكل التالي:



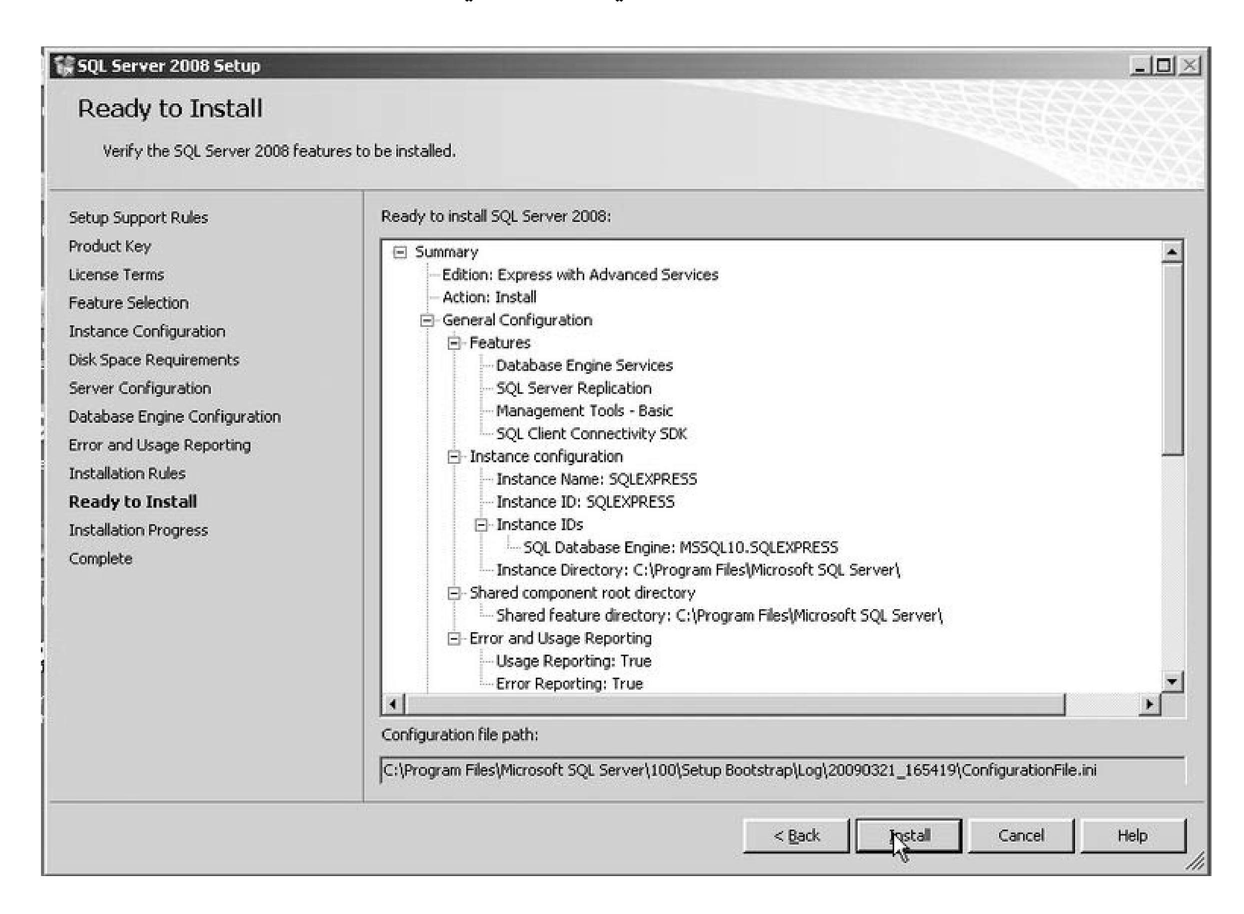
والشكل التالي:



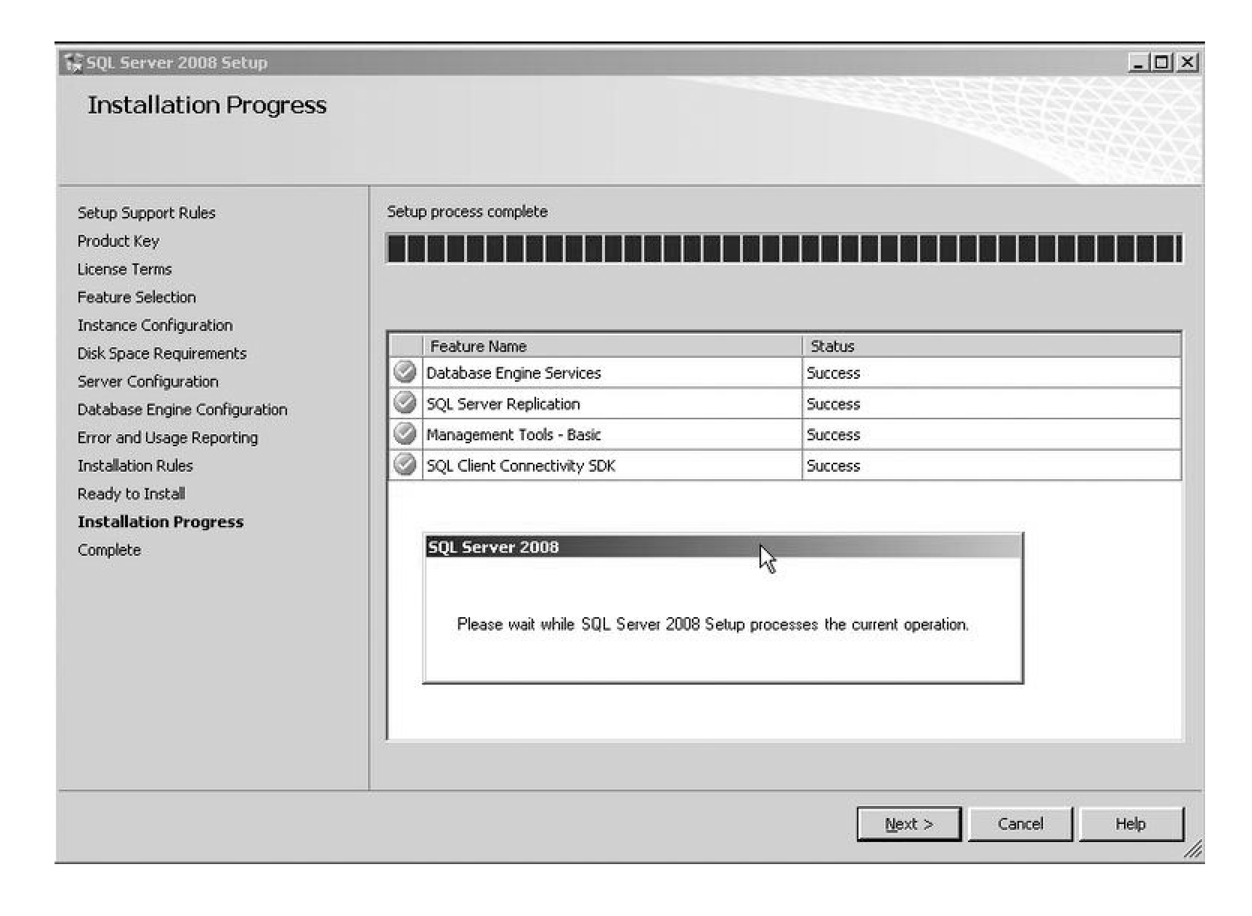
بعد الضغط على Next نذهب إلى شاشة تقارير الإستخدام والأخطاء Error and Using Reporting فنقوم بإختار الخيارات كلها ، لأن ذلك يساهم في دعم المنتج المستمر لتلافي أية مشاكل مستقبلية ، ثم بعد ذلك عند الضغط على Next نرجع إلى شاشة قواعد التثبيت كما في الشكل التالي:



عند الضغط على Next نذهب إلى شاشة الإستعداد للتثبيت كما في الشكل التالي:



عند ذلك تبدء عمليه التنصيب بالمواصفات التي حددناها، و تأخذ العملية بعض الوقت إلى ان تتم كما في الشكل التالي:



والشكل التالي أيضاً :

SQL Server 2008 Setup

Complete

Your SQL Server 2008 installation completed successfully.

Setup Support Rules
Product Key
License Terms
Feature Selection
Instance Configuration
Disk Space Requirements
Server Configuration
Database Engine Configuration
Error and Usage Reporting
Installation Rules
Ready to Install
Installation Progress
Complete

Summary log file has been saved to the following location:

C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Log\20090321_165419\Summary_Oracle_20090321_165419.txt

Information about the Setup operation or possible next steps:

Your SQL Server 2008 installation completed successfully.

Supplemental Information:

The following notes apply to this release of SQL Server only.

Microsoft Update

For information about how to use Microsoft Update to identify updates for SQL Server 2008, see the Microsoft Update Web site <http://go.microsoft.com/fwlink/?LinkId=108409> at http://go.microsoft.com/fwlink/?LinkId=108409.

Reporting Services

The Reporting Services installation options that you specified in Setup determine whether additional configuration

Close Help

اضغط close في الشكل السابق ، ثم Finish لإنهاء التثبيت بنجاح كمال بالشكل التالي:



وبهذا نكون قد أنهينا عملية التثبيت بنجاح ، للنسخة Express ، والتي لن تختلف كثيراً عن النسخ المحترفة ، يُمكنك أيضاً أن تقوم بتثبيت Visual Studio 2010 بأي نسخة حسب اختيارك ، ويمكنك الاستعانة بالبحث حال عدم توفر أي من هذه النسخ عندك لأني افترض فيك الجدية كمبرمج لتعتمد على نفسك حال مواجهة أية مشاكل في تثبيت منتج أو عدم موائمته مع إمكانيات جهازك او خادمك او البحث عنه للحصول عليه.
يتبقى الجزء اليسير وهو تثبيت قاعدتين البيانات اللتين ستتعامل معهما خلال الكتاب وهما القاعدتان المشهورتان Northwind و AdventureWorks يُمكنك ان تحصل قاعدة البيانات **AdventureWorks** من هذا الرابط :
http://msftdbprodsamples.codeplex.com/releases/view/37109
ومنه بعد التحميل اتبع الخطوات البسيطة لتثبيتهما لكن ليس قبل ان تثبت خادم قواعد البيانات SQL Server 2008 Express وهذا الرابط لقاعدة البيانات **Northwind** :
http://northwinddatabase.codeplex.com/
بالضغط على زر Download الموجود في يمين الصفحة ، ثم اتبع خطوات تثبيتها أيضاً من خلال وثائق الرابط نفسه او بالبحث عن تلك الكيفية (لم اشرع في شرح ذلك حتى تتعود الاعتماد على نفسك لأنك في طريقك لعالم البرمجة الأكثر احترافاً).

بالمتابعة في بداية الفصول القادمة ستتمكن من التعامل مع هذه البيئات الرائعة لتبدأ انطلاقتك نحو عالم برمجة قواعد البيانات.

الفصل الأول

مدخل إلى قواعد البيانات العلائقية Relational Databases

فى هذا الفصل سوف تتعلم الآتى:

- ما هى قواعد البيانات ؟
- متى أختار بين برامج الجداول الحسابية وبين قواعد البيانات ؟
- لماذا نستخدم قواعد البيانات ؟!
- فوائد استخدام نظم إدارة قواعد البيانات العلائقية RDBMS ؟
- الفرق بين RDBMS لأنظمة سطح المكتب و الخوادم Servers
- دورة حياة نظم قواعد البيانات DB Life Cycle
- إيجاد العلاقات بين البيانات
- فهم المفاتيح Keys
- فهم تمامية البيانات Data Integrity
- مبادئ الـ Normalization
- مساوئ الـNormalization

ما هى قواعد البيانات ؟

بمنتهى البساطة فى تعريف قواعد البيانات نقول أنها مجموعة من البيانات المهيكلة **structured** أى موضوعة وفقا لمنظومة معينة ،فالغرض الأساسى لأى قاعدة بيانات هو تنظيم معلومات كبيرة الحجم تيسيرا على المستخدم حال قيامه بعملية استعلام أو تعديل أو إضافة لهذه المعلومات .

ما هو نظام إدارة قواعد البيانات Data Base Management System (DBMS) ؟

هو عبارة عن برنامج لإدارة قواعد البيانات وإنشائها والتعديل فيها ، أى هو أداة المستخدم فى فعل ما يشاء فى أى بيانات على حسب إمكانيات البرنامج ، فمثلا يُمكن هذا البرنامج المستخدم من إجراء الاستعلامات (ستعرفها مامعنى الاستعلام لاحقاً) –الخاصة بإرجاع البيانات وعرضها فى جداول ،أو التعديل عليها ....إلخ من العمليات .

ما هو الـ Relational DBMS ؟

يستطيع الـ **DBMS** من التعامل مع البيانات فى صورة جداول (صفوف أو سجلات –أعمدة أو حقول) تشبه تلكم الجداول فى البرامج المحاسبية مثل **Excel** وهى صورة سهلة ومنطقية لتنظيم البيانات ، ومن هنا ظهرت قواعد البيانات العلائقية أو **RDBMS** التى هى مربط الفرس الآن فى التعامل مع البيانات فى معظم التطبيقات سواءاً أكانت تطبيقات سطح المكتب أو ويب أو حتى أجهزة كفية .

متى تختار بين قواعد البيانات والجداول الممتدة Spreadsheets ؟

يأتى سؤال بديهى..بما أن قواعد البيانات تشبه فى عملها الجداول الممتدة **Spreadsheets** فلماذا إذا ظهرت ؟!! ....سؤال جيد ، والإجابة تكمن فى المرونة التى يوفرها لك أى نظام إدارة لقواعد البيانات من استرجاع سلس وسهل للبيانات ، إجراء العمليات على هذه البيانات مهما كانت مفرقة فى الجداول ،إجراء عمليات على الجداول دفعة واحدة دون تجزئتها (كل هذا سيظهر لك لاحقا لاتقلق من المصطلحات إن لم تكن تعرفها) .

لماذا أستخدم قواعد البيانات ؟

توفر لى قواعد البيانات الحلول للعديد من المشكلات التى تقابلنى فى الحياة مثال :

- **التوثيق للبيانات** الكبيرة الحجم والمبعثرة (كالإرشيف مثلا فى المؤسسات المختلفة) بدلا من الطرق التقليدية فى الحفظ كالورق والملفات .
- **السرعة** فى جلب المعلومة مهما كان حجم البيانات عندى ،أسرع بكثير حتى ولو كان النظام الورقى التقليدى عندى مرتب بأى شكل كان .
- **توفير الوقت والمجهود** المبذولان فى ترتيب وتنظيم البيانات بالطرق التقليدية والتى بدورها معرضة لأخطاء كثيرة وجسيمة .
- **الإعتمادية فى جلب المعلومة** ،فالبشر يخطئ نتيجة للضغوط اليومية فى العمل ،وبالتالى وقت الازمات تظهر الحاجة إلى معلومات وبأقصى سرعة ،بطبيعة الحال لن تجد اسرع من البيانات المميكنة فى صورة قواعد بيانات وفاءاً بطلبك فى زمن صارت فيه للثانية قيمة .

فوائد استخدام نظم إدارة قواعد البيانات العلائقية RDBMS ؟

تستطيع **RDBMS** من تحقيق فوائد جمة عن طريق التحكم فى الآتى :

- **التكرارية Redundancy** :فهى تمنع أى تكرار سواءا مقصود او غير مقصود لبيانات موجودة مسبقا وبالتالى تتحكم فى توفير وعدم إهدار المساحة على القرص الصلب او أى وحدة تخزين اخرى.

- **تضارب البيانات Inconsistency** :من الفائدة السابقة نحقق ضمان عدم تضارب البيانات ودقتها فمثال ،لو أنك أدخلت اسم نفس الشخص الرباعى مرتين مثلا فهناك احتمال للخطأ فى أحدهما مع أنه نفس الشخص مما يوحى بوجود شخصين مختلفين ،فالتحكم فى تكرار البيانات وادخالها يلغى هذه المشكلة من الأساس .
- **تكامل البيانات Data Integrity**: فالنظام يحقق نوع من الانسجام بين البيانات يمكن من خلاله استخراج معلومة صحيحة (سيتم مناقشة قضية تكامل البيانات تلك لاحقا) .
- **تدارك الخطأ** :فى حالات فشل اتمام أى عملية كتحويل الأموال مثلا أو الولوج إلى أنظمة سرية مثلا أو حتى العمليات المزدوجة كإجراء تعديل من قبل شخصين على نفس البيانات فى نفس الوقت هناك خط دفاعى لتدارك هذه المشكلات عن طريق RDBMS.
- **تأمين البيانات** : ليس كل شخص يتعامل مع البيانات مخول له القيام بعمليات كاملة على البيانات ،فهناك أشخاص لعرض البيانات فقط ،وآخرون لإجراء تعديل عليها ،وآخرون لديهم كافة الصلاحيات من حذف لهذه البيانات والتعديل ....إلخ من العمليات .
- **العمليات المنتظمة واسترجاع الأخطاء:** فى بعض الحالات يكون هناك تسلسل لعمليات مختلفة على قواعد البيانات يستطيع **RDBMS** من هذا ،فضلا عن استرجاعه لهذه العمليات حال حدوث خطأ مفاجئ لا قدر الله.
- **تنظيم التخزين:** يمكنك **RDBMS** من تنظيم للبيانات المخزنة على وحدات التخزين المختلفة بميكانيكية تسهل عمليات الاسترجاع والبحث عن طريق مايسمى بـ**Internal Schema** دون تدخل منك .

الفرق بين RDBMS لأنظمة سطح المكتب و الخوادم Servers

تنقسم صناعة البرمجيات اليوم المتعلقة بقواعد البيانات تقريبا إلى قسمين رئيسيين :

- **قواعد بيانات أنظمة سطح المكتب او PCs .**
- **قواعد بيانات أنظمة الخوادم Servers.**

فيما يلى الفروق الجوهرية بين النوعين :

قواعد بيانات أنظمة سطح المكتب او PCs

تتسم قواعد بيانات أنظمة سطح المكتب او PCs بقلة عدد المستخدمين او بالأحرى المنتفعين من قاعدة البيانات عن طريق أنظمة رخيصة نسبيا مثل الـ (**MS Access –Lotus-FoxPro-SQL Server Express**)

وفى الغالب لا تتضمن هذه الانظمة العلميات المعقدة او الإمكانات الكبيرة نظراً لعدم الاحتياج إليها على أنظمة سطح المكتب فهى تلائم المؤسسات الصغيرة والمتوسطة الحجم بمنتهى الفاعلية ، لكنها تختلف عن أنظمة الخوادم فى الآتى :

- **أرخص كثيرا :** فبالقليل من المال تستطيع اقتناء رخصة لحزمة كاملة كالأوفيس متضمنة برنامج الأكسس وبالتالى أنت والمستخدم النهائى فى غنى عن المساءلة القانونية عن رخصة نظام قواعد البيانات الملحق ببرامجك وهذا الأمر هام جدا لمن عمل فى تصنيع برامج قواعد البيانات فى السوق .
- **سهولة الاستخدام :** فقط أنت تخطط لبرنامجك ثم ما عليك إلا أن تفتح برنامج ذو واجهة رسومية **GUI** سهلة كالأكسس مثلا وتبدأ فى التنفيذ ، دون الحاجة إلى إجراء استعلامات **SQL** أو أى طرق أخرى صعبة او معقدة .

قواعد بيانات أنظمة الخوادم Servers

على العكس من أنظمة قواعد البيانات الخاصة بسطح المكتب ، تتسم قواعد بيانات الـ Servers بالتعامل مع كم كبير ومعقد من البيانات دفعة واحدة ،ليس هذا فحسب بل ومن أكثر من مستخدم فى نفس الوقت دون أى خلل وهذا يرجع إلى طبيعة الخوادم وإماكانياتها الكبيرة مقارنة بالأجهزة المنزلية العادية وكمثال على قواعد بيانات الخوادم هناك العملاق **أوراكل** وكذا **MS SQL Server** وأنظمة شركة **IBM DB2** و**Sybase** .

ويكمن الإختلاف الجوهرى بين هذه الانظمة وسابقتها فى الآتى :

- **المرونة :** وحقيقة هى من أهم ميزاتها ،فقد تم تصميم هذه الأنظمة لتلائم وبمنتهى المرونة أنظمة التشغيل المختلفة **كالويندوز واللينكس واليونكس** ولتتلقى العديد من الاستعلامات فى ذات الوقت وتتعامل معها بمنتهى السهولة والسرعة أيضا .

- **الإعتمادية :** توفر قواعد بيانات أنظمة الخوادم Servers القدرة على الاعتماد عليها بدرجة ٢٤ ساعة طيلة ٧ أيام متصلة طبقا لحاجة السوق إلى ذلك ..مثال هذا أنظمة البنوك والشركات العملاقة **كميكروسوفت** وكموقع كبير **كأمازون** مثلاً فهى عادة ماتلحق ببعض الميزات مثل الـ Mirroring والـ Log Shipping .
- **سرعة الأداء :** لأنها تعمل على اجهزة الخوادم فهى تتسم بالسرعة العالية فى الاستجابة للأوامر والعمليات المختلفة ، فأجهزة الخادم دائما ما تلحق بحجم كبير من الذاكرة وسعات التخزين ، مما لاشك فيه يؤثر بالإيجاب على السرعة والدقة المطلوبة .
- **التمدد :** من الميزات المهمة جدا ميزة التمدد والاستعداد الدائم للطوارئ والزيادات فى أى وقت فى حجم البيانات وكثافتها ، نخيل مثلا لو أن بنك فى اليابان كان حجم تعاملاته اليومية ٢٠ مليون عملية تم دمجه مع بنك آخر حجم تعاملاته ١٠ مليون عملية ، إن لم تكن قواعد البيانات مهيأة تماما لمثل هذه الإجراءات الطارئه ...فالخسائر ستكون فادحة فى العملاء .

دورة حياة نظم قواعد البيانات DB Life Cycle

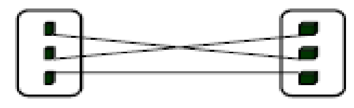
كما فى عالم النبات والحيوان دورة حياة ، أيضا فى عالم البرمجيات دورة حياة للمشاريع ابتداءا من التصور وانتهاءا بمراحل كالتوزيع واصدار الترقيعات ....إلخ .
كذا فى حالة نظم قواعد البيانات يبدأ التطوير بالفكرة ثم التنفيذ ،والذى بدوره ينقسم إلى عدة مراحل ، لايتم الانتقال إلى مرحلة إلا بعد تجاوز المرحلة السابقة لها **(Block by Block) .**
قبل الشروع فى تصميم اى نظام ،لابد وأنك تعمل وفقا لنموذج قياسى معين Model ،والذى بدوره يحوى كل الخطوات اللازمة لبدأ تنفيذ فكرتك البرمجية وبالتالى فلن يواجه فريق التطوير أى مشاكل تعترضه من تداخل فى الأفكار أو العشوائية فى التنفيذ وضمان جودة برمجية عالية .
تنقسم دورة حياة نظم قواعد البيانات إلى عدة مراحل ،بدءاً من الـ **global schema** وانتهاءا بالتنفيذ والصيانة **maintenance** :

- **تحليل المتطلبات Requirement analysis:** قبل الشروع فى التصميم لابد وان أعى المشكلة المراد حلها بقواعد البيانات جيدا ،يتطلب هذا عدة لقاءات مع المستخدمين أو الموظفين من خلالها يُعرف كيف يدار النظام ، ومن أين وإلى أين تتدفق البيانات بهذه الطريقة تضمن توافق تنفيذك لمشروعك مع متطلبات العميل (هذه المرحلة بحق هى عصب أى مشروع برمجى).
- **التصميم المنطقى Logical design:** ياتى بعد مرحلة جمع المتطلبات ،مرحلة تصميم كروكى لما ستكون عليه البيانات ،فابستخدام العلاقات والنماذج مثل ER diagrams نستطيع توضيح هذه العلاقات والترابطات بين البيانات.
- **التصميم الحقيقى Physical design:** متى تم الانتهاء من التصميم المنطقى ،تأتى هذه المرحلة الهامة وهى وضع الجداول وإختيار المفهرسات **Indexers** لإكمال البنية الهيكلية لقاعدة البيانات.
- **مرحلة بناء قاعدة البيانات :** هنا يبدأ المجهود السابق يثمر عن قاعدة البيانات الحقيقة التى ستستخدم فى مشروعك مستخدمين نظام إدارة قواعد البيانات العلائقية **RDBMS (فى الحقيقة سنستخدم مايعرف بـ DDL وهى إختصار لـ Data Definition Language ستعرف هذا لاحقا إن شاء الله)**
- **مرحلة التعديل على البيانات Data modification :** باستخدام لغة التعديل على البيانات **Data Modification Language او DML** تستطيع إجراء الإستعلامات وإنشاء المفهرسات وتحديث قاعدة البيانات ووضع القيود مثل **التكامل المرجعى Referential Integrity .**
- **مراقبة قاعدة البيانات Database Monitoring:** وتعتبر هذه المرحلة هامة جدا بعد عمليات التنفيذ السابقة ،ضماناً لتلاقى التنفيذ مع المتطلبات المنشودة ،ففى حال وجود مشاكل أو ظهور خطأ ما فى التصميم عن طريق المراقبة تستطيع تلافى هذا الخطأ بالرجوع إلى الخطوات السابقة وإجراء التعديل اللازم .وهكذا دواليك تستمر دورة حياة قاعدة البيانات بالوصول إلى هذه المرحلة ثم العودة ثانية إلى المراحل السابقة إذا لزم الأمر .

إيجاد العلاقات بين البيانات

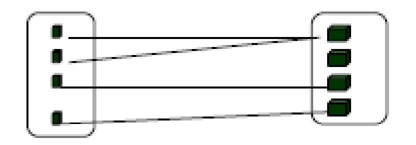
تعتبر الجداول الشكل المبدئ لقواعد البيانات العلائقية ،فى الواقع يتم تخزين البيانات وعلاقاتها معا فى قاعدة البيانات فى جداول.
تتكون الجداول من صفوف وأعمدة ، كل عمود يُعبر عن معلومة جزئية .
فى قواعد البيانات لابد وأن ترتبط الحقول أو أى جزئية من البيانات مع بعضها البعض بعلاقة ما .
فمثلا قسم للمشتريات فى شركة ما ، هناك موردون لهذه الشركة وبأصناف شتى نحتاج عند تنفيذ قاعدة بيانات لذلك القسم إلى عدة علاقات موضحة فى التالى :

- العلاقة **One-to-One:** تعنى أن قيمة واحدة فى الجدول أ مثلا تقابلها قيمة واحدة (صف) فى الجدول **ب** انظر الرسم :



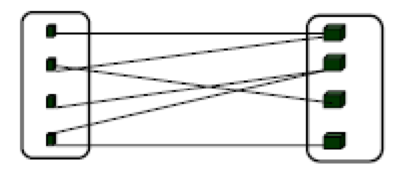
فمثلا لكل قسم يوجد مدير واحد فقط ..لذا يتم ربط جدول المديرين بجدول الأقسام عن طريق العلاقة **(١:١).**

- العلاقة **One-to-Many :** تعنى أن قيمة فى الجدول الأول تقابلها قيمة فى الجدول الثانى وهذه القيمة بدورها يقابلها قيمة أو أكثر فى الجدول الأول ...



مثال على ذلك طلبات الزبائن على سلع أو منتجات معينة ..فالزبون الواحد مخول له شراء عدة منتجات مختلفة وكذا نفس العلاقة من السلع للزبائن أى أكثر من زبون يحصل على نفس السلعة .

- العلاقة **Many-to-Many :** تعنى أن قيمة فى الجدول الأول تقابلها قيمة أو أكثر فى الجدول الثانى والعكس صحيح ...



فمثلا فى جدول الطلاب يستطيع الطالب التسجيل فى أكثر من مادة ، وجدول المواد الدراسية يستطيع أن يسجل العديد من الطلاب للمادة أو أكثر .
فى الغالب عند تنفيذ هذه العلاقة يكون هناك جدول ثالث للربط بين الجدولين لأن هذا النوع من العلاقات معقد نوعا ما .

فهم المفاتيح Keys

عرفنا أن تمثيل العلاقات يتم بين البيانات فى الجداول وفقا لطبيعة العلاقة ، ولإيجاد هذه العلاقات لابد من وجود صفوف أو حقول مثلا للربط بين هذه الجداول ، وأمثل علاقة بين هذه الجداول هى المفاتيح **Keys**،هذه المفاتيح ماهى إلا أعمدة متشابهة بين الجداول وذلك لتمييز الصفوف بعضها عن بعض .
أى **RDBMS** يتعامل مع نوعين رئيسيين من المفاتيح : **Primary Key (P.K) , Foreign Key (F.K) :**

١. المفتاح الرئيسي P.K:
هو عمود او أكثر ، يتصف بأنه غير حاوى لبيانات متكررة أى بيانات فريدة **Unique** مميزا به كل صف فى الجدول .تنبه إلى الآتى عندما تختار مفتاحا رئيسيا **P.K** :

- كل سجل فى الجدول لابد ألا يحتوى على قيمة خالية **NULL** .
- كل قيمة مدخلة إلى العمود الحاوى للمفتاح لابد أن تكون قيمة فريدة كما أسلفنا.
- ضمان الحفاظ على القيمة الموضوعة فى الجدول من عدم التغيير مستقبلا أثناء اجراء العمليات على قاعدة البيانات .
- يوجد فى الجدول مفتاح رئيسي واحد فقط .

إلى جانب ضمان المفتاح الرئيسى **P.K** لفرادة البيانات الموجودة فى كل سجل ، فهو أيضا يسهل عملية البحث داخل الجدول وذلك لأنك بمجرد انشاءك للمفتاح الرئيسى يتولد اتوماتيكيا فهرس للجدول يسهل عليك عملية البحث وبالطبع هذا كله يعود إلى الـ **RDBMS** .
ليس هذا فحسب بل من الممكن أخذ أكثر من **Entity** وإعتبارها مفتاح رئيسى وذلك تحقيقا لمبدأ الفرادة **Uniqueness** ويسمى هذا،كما أنه يمكن ترشيح اى مفتاح ليصبح مفتاحا رئيسيا (وفقا للصفات السابقة الذكر) ويسمى هذا **المفتاح المرشح Candidate Key** .

٢. المفتاح الخارجى Foreign Key :
المفتاح الخارجى **F.K** عبارة عن صفة مرجعية للمفاتيح الرئيسية ..أو قيمة غير مكررة **(Unique)** فى جدول آخر وهى هامة جدا فى تحقيق خاصية التكامل المرجعى والتنقل بين عناصر الجداول المترابطة ..انظر الشكل للتقريب ..ولاتنس ان الأمر مجرد تنشيط للذهن ومراجعة لما تعرفه سابقا .

Student				
St _No	St _Name	Gpa	Birth Date	Dept Code
2001-01-10	Sami	3.75	01-01-1981	Comp
2001-02-99	Khalid	3.5	10-10-1982	Math
2000-01-101	Ali	4.2	12-08-1980	Comp

Department	
Dept_Code	Dept_name
Comp	Computer
Math	Mathematics

لاحظ أن الـ **Dept_Code** فى الجدول الأصغر مفتاح رئيسى ..وفى الجدول الأكبر مفتاح خارجى **F.K.**

فهم تمامية البيانات Data Integrity

وتعنى ضمان صحة قيم بيانات قاعدة البيانات وكذا ارتباطها معا .وهى تنقسم إلى قسمين **Entity Integrity** و **Referential Integrity.**

- **أولاً: Entity Integrity**

ذكرنا فى الشرط الأول من شروط اختيار المفتاح الرئيسى ألا تكون القيمة خالية **Null** وهذا لضمان وجود قيمة على طول المفتاح الرئيسى فى كافة الصفوف ، هذا الضمان هو مايعرف بـ **Entity Integrity**،حيث يقوم الـ **DBMS** بمنع إجراء استعلامات **(Insert-Update)** إلى المفتاح الرئيسى متضمنة قيمة مكررة أو غير فريدة .

- **ثانياً : Referential Integrity**

عندما يتم انشاء الجداول وإيجاد العلاقات بينها فى قاعدة البيانات ،يتم اختيار المفاتيح الخارجية **F.K** والتى عن طريقها يتم إدارة العلاقات بين الجداول تحقيقا لمبدء التكامل المرجعى **R.I**، فهو يضمن أن كل قيمة فى المفتاح الخارجى **F.K** تقابلها قيمة فى جدول آخر للمفتاح الرئيسى **P.K** .

فهم مبادئ الـ Normalization

الـ **Normalization** باختصار شديد هى منع تكرار البيانات أو الزج ببيانات ليس لها داع فى أثناء قيامك بالتصميم المنطقى لقاعدة البيانات ،وبهذه الطريقة توفر على نفسك عناء التحديث المستقبلى لقاعدة البيانات ، فضلا عن سرعة الاداء بعد التنفيذ الفعلى لقاعدة البيانات .بالطبع عملية كتلك تمر بعدة مراحل إلى ان تصل الى النموذج المعيارى **3NF Third Normal form** وهذه المراحل إذا اردنا تشبيها بليغا لها فهى مثل عملية الغسيل ثم الشطف ثم الشطف النهائى ☺ ، يمكنك الاستزادة بقراءة المزيد عنها هنا http://en.wikipedia.org/wiki/Database_normalization.

مساوئ الـNormalization

بطبيق مبادئ الـ **Normalization** يصبح عندنا مجموعة كبيرة من الجداول وكذا العمليات الوصلية(**Joins**) لإسترجاع البيانات، وبما أن البيانات موزعة على الجداول فيما بينها ،فإن مثل هذه العلميات تكون معقدة وبالتالى ترهق عملية المعالجة وتستنزف وقتها ،فمن الممكن وقتها نتخلى عن بعض مبادئ الـ **Normalization** فى سبيل التخفيف عن المعالجة بعض الشئ ،تذكر أن الغرض من هذه العلميات ماهو إلا الوصول إلى التصميم الأمثل لقاعدة البيانات تجنبا لمشاكل الأداء والتحديث المستقبلى .

خاتمة الفصل

تذكر أن الهدف من هذا الفصل كان تنشيط الذاكرة ليس إلا ببعض ما درسته أنت فى السابق سواء مع نفسك أو فى فصول الجامعة ، فإذا وجدت انها ثقيلة عليك نوعا ما ومملة ...لا تقلق أبدا فهذا لن يعيقك لتكون مبرمج قواعد بيانات محترف ..وسوف تكتسب كل هذا بالخبرة إن شاء الله ...فقط تابع المسير فالقادم أسهل بكثير .

الفصل الثانى

كتابة استعلامات قواعد البيانات Database Queries

فى هذا الفصل ستتعلم

كتابة الاستعلامات داخل SQL Server 2008، والتعرف على اللغة الداخلية T-SQL المسئولة عن كتابة هذه الاستعلامات ،بالاستعانة بقاعدة بيانات شهيرة مثل Northwind أو Adventure Works.
بإذن الله سيغطى هذا الفصل الآتى ...

- المقارنة بين QBE and SQL
- التعرف على SQL Server Management Studio Express
- التعرف على الاستعلامات Queries
- اجراءات وأوامر التعامل مع الجداول
- جملة GROUP BY
- المعامل PIVOT
- دالة ROW_NUMBER()
- جملة PARTITION BY
- فهم الـ Pattern Matching
- دوال التجميع Aggregate functions
- دوال الوقت والتاريخ DATETIME functions
- JOINS

المقارنة بين QBE and SQL

لكى نتعامل مع قواعد البيانات العلائقية هناك نوعان من اللغات للتواصل بين المستخدم وبين قاعدة البيانات وهما (QBE- SQL).

QBE: هى اختصار للجملة Query By Example وهى طريقة لاجراء الاستعلامات على قواعد البيانات معتمدة على الواجهه الرسومية معتمدة على مبدأ (Point and Click).لقد تم تطوير QBE على يد موشى زلووف فى معامل IBM بالتزامن مع تطوير SQL فى منتصف السبعينات ، وهى تختلف عن SQL فى سهولة اجراء الاستعلامات معتمدين كما قلت على الواجهة الرسومية فقط ارسم الجدول وقم بالاشارة على ماتريد استخلاصة من بيانات او حذف او تعديل بيانات وسيستجيب لك نظام ادارة قواعد البيانات ،وهى طريقة تعتبر مثالية فى حالة قواعد البيانات الغير معقدة والتى يتم تمثيلها بالقليل من الجداول.
وعلى الرغم من انها تطوير شركة IBM إلا أن شركات كميكروسوفت قامت باجراء بعض التعديلات عليها واستخدامها فى قواعد بيانات اكسس وتظهر جليا عند اجراء استعلامات على الـForms .

SQL: هى اختصار للجملة Structured Query Language، وهى لغة تم تطويرها فى معامل IBM ايضا من قبل مجموعة تطوير فى مركز ابحاث سان جوز- شامبرالن وريموند - وهى أساسا قد تم تطويرها للتعامل مع نظام قواعد بيانات يسمى System R وهو نظام مبنى على فرضيات Codd ، ولمن لايعرف فان Codd هذا يعتبر الأب الشرعى لقواعد البيانات العلائقية بعد نشره ورقة بحث يوضح فيها الأسس والأطر لهذا النظام الجديد.
فى عام ١٩٨٦ تم اعتماد لغة SQL من قبل ANSI وأيضا من قبل ISO عام ١٩٨٧ وتم نشر اللغة على وضعها القياسى هذا تحت اسم SQL1 .ومن هذه الانطلاقة و سكول آخذة فى التطوير ففى عام ١٩٨٩ وعام ١٩٩٢ تم عمل نقلة نوعية فى اللغة ثم مع العام ١٩٩٩ واصدار SQL3 التى دعمت مميزات كالأهداف الموجهه والتى كانت نواه لقواعد البيانات الموجهه بالكائنات Object Relational DB.
وعلى الرغم من وضعها SQL كمعيار لكن هناك شركات مثل اوراكل وميكروسوفت اصدرت انتاجها الخاص من SQL تيسيرا لبعض المهام على قواعد البيانات الخاصة بها ، وفى كافة الاحوال فهذا لايختلف كثيرا عن الاصدار المعيارى من SQL ومثال على ذلك T-SQL المستخدمة فى هذا الكتاب ، ربما لن تعمل اذا حاولت اجراءها على قواعد بيانات اخرى بخلاف SQL –Server.

<u>تنوية: أعلم أن رؤوسكم قد تصدعت من هذه النبذة التاريخية ولكن هناك فوائد كثيرة ستخرج منها من هذه النبذة وهى ايضا لايقاف بعض الجدل حول لغة SQL .</u>

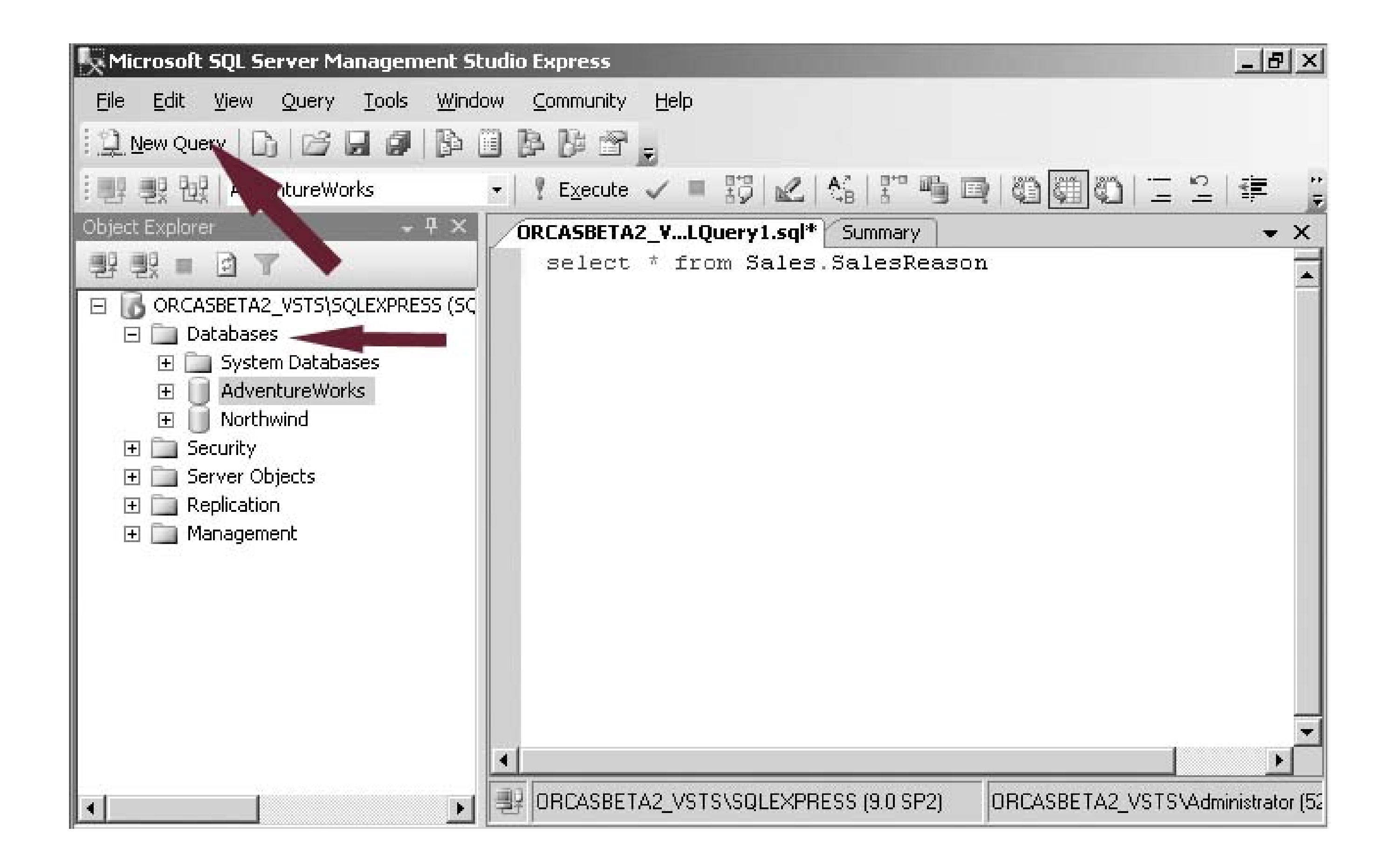
البدء مع كتابة الاستعلامات Queries

كتعريف مبسط للإستعلام : هو عملية استخلاص للمعلومات من قاعدة البيانات.يستلزم ذلك وجود منصة أو نافذة لكتابة هذه الاوامر او الاستعلامات لكى نستخلص البيانات من قاعدة البيانات.
هيا بنا نشمر عن ساعدينا ونبدأ بكتابة أول استعلام لنا ، لكن قبل ذلك تأكد من وجود SQL Server Management Studio Express والذى سنشير له دوما بـ SSMSE (راجع الملحق لكى تتعلم كيفية تثبيته والتعامل مع نوافذه).

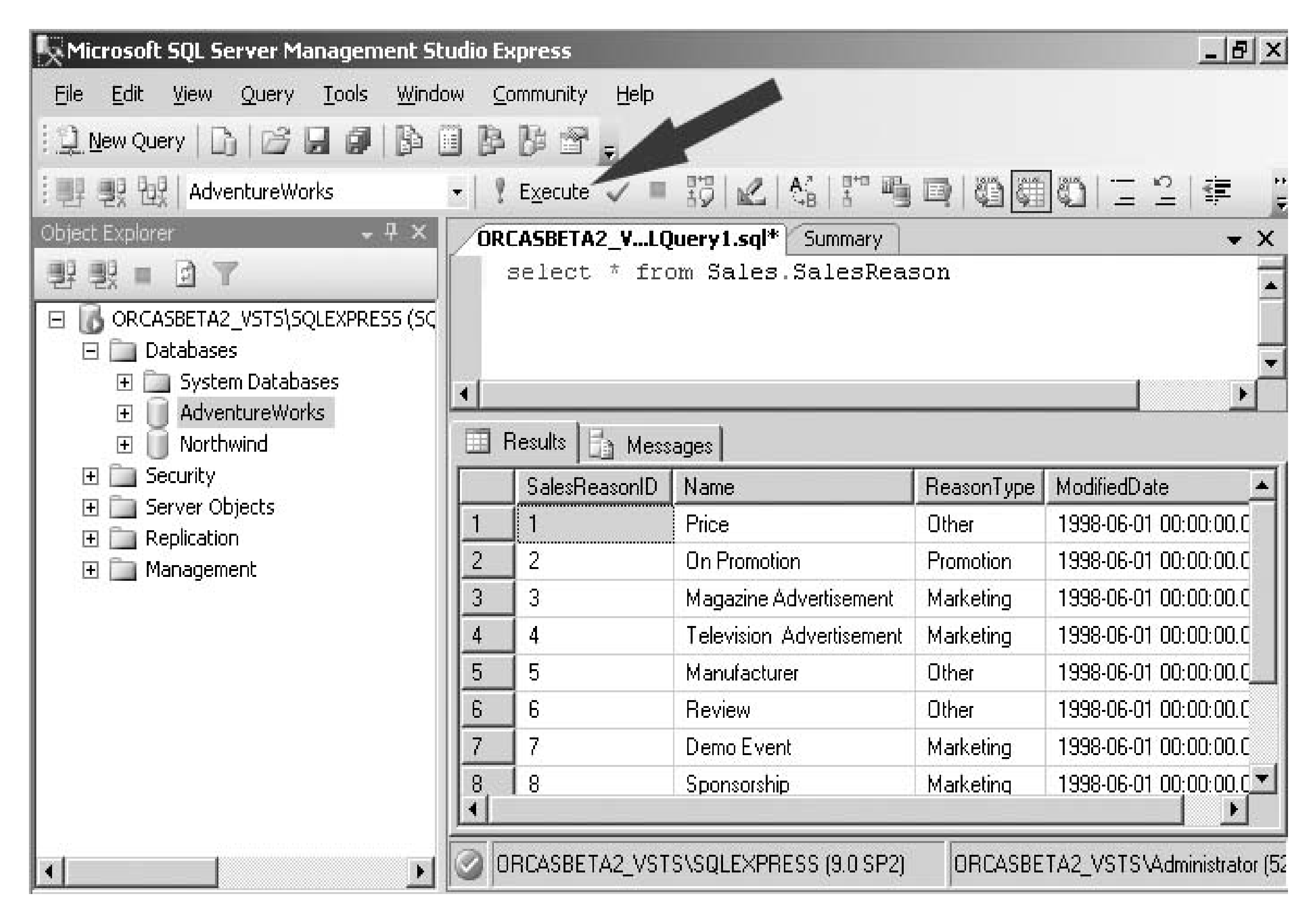
جرب هذه: تنفيذ استعلام بسيط

١- افتح SSMSE ثم قم بتمديد شجرة الـ Databases واختر Adventure Works database.
٢- اضغط على الزر New Query كما بالشكل واكتب هذا الاستعلام بداخل النافذة التى ستظهر لك

Select * from Sales.SalesReason



٣- اضغط على زر Execute او اضغط F5 او اختر قائمة Query -> Execute ليظهر ناتج الاستعلام كما بالشكل



لشرح ماحدث ببساطة فإن النجمة * مع كلمة SELECT تشير الى ارجاع كافة الأعمدة من الجدول المراد عمل استرجاع لبياناته.

اجراءات وأوامر التعامل مع الجداول

والتى سنشير اليها بـ CTE او Common Table Expressions وهى خاصية جديدة تم اضافتها الى SQL Server 2008 وهى ببساطة حالة مؤقتة من اجراءك لاستعلام معين مثل SELECT , INSERT,DELET UPDATE وتظهر ميزتها جليا حين تقوم بعمل استعلام ما على جدول تم اشتقاقه بدلا من استخلاصها فى جدول مؤقت ثم اجراء الاستعلامات عليه ثم حذفه ، كان هذا فى السابق مضيعة للموارد والوقت.

تتكون الـ CTE من ثلاثة أجزاء رئيسية

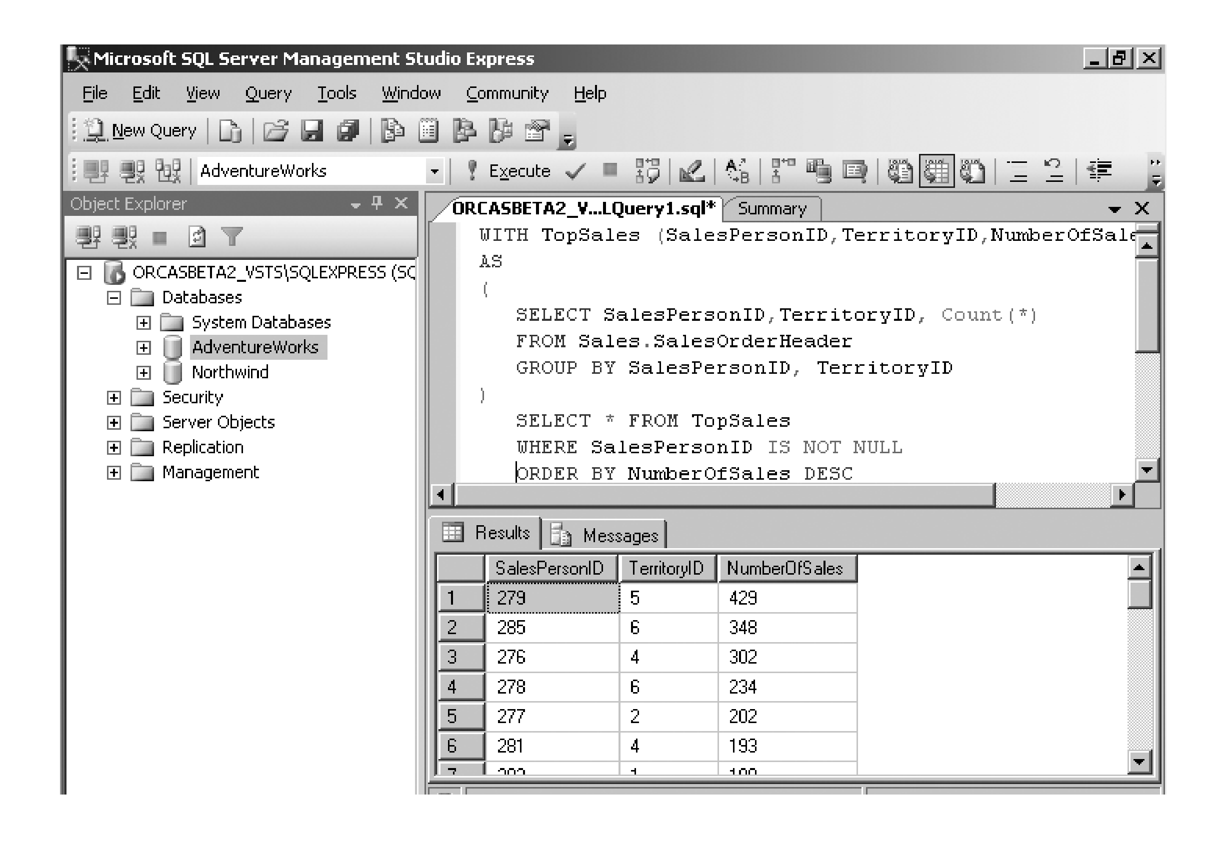
١- اسم الـ CTE ملحقا بكلمة .WITH
٢- اسم العمود (اختيارى)
٣- الاستعلام الذى يظهر بين القوسين ( ) بعد كلمة AS

مثال يشرح ذلك

افتح SSMSE ثم اكتب هذا الاستعلام :

```
WITH TopSales (SalesPersonID,TerritoryID,NumberOfSales)
AS
(
SELECT SalesPersonID,TerritoryID, Count(*)
FROM Sales.SalesOrderHeader
GROUP BY SalesPersonID, TerritoryID
)
SELECT * FROM TopSales
WHERE SalesPersonID IS NOT NULL
ORDER BY NumberOfSales DESC
```

بعد ذلك اضغط F5 وسترى النتيجة كما بالشكل



فى هذا المثال الذى حددنا اسم CTE له وكذا الاعمدة الثلاثة (SalesPersonID,TerritoryID,NumberOfSales) نلاحظ ان جملة SELECT SalesPersonID,TerritoryID, Count(*)
هى الأخرى ستعود بناتج ثلاثة أعمدة والعمود المنفرد المخصص من SELECT list سيشير الى الأعمدة التى تم تخصيصها فى توصيف الـ CTE .

يبدوا الامر معقدا بعض الشئ ☺ لكن لابأس هذا الفصل لاينبغى عليك فهم الكثير بقدر تطبيق الكثير

جملة GROUP BY

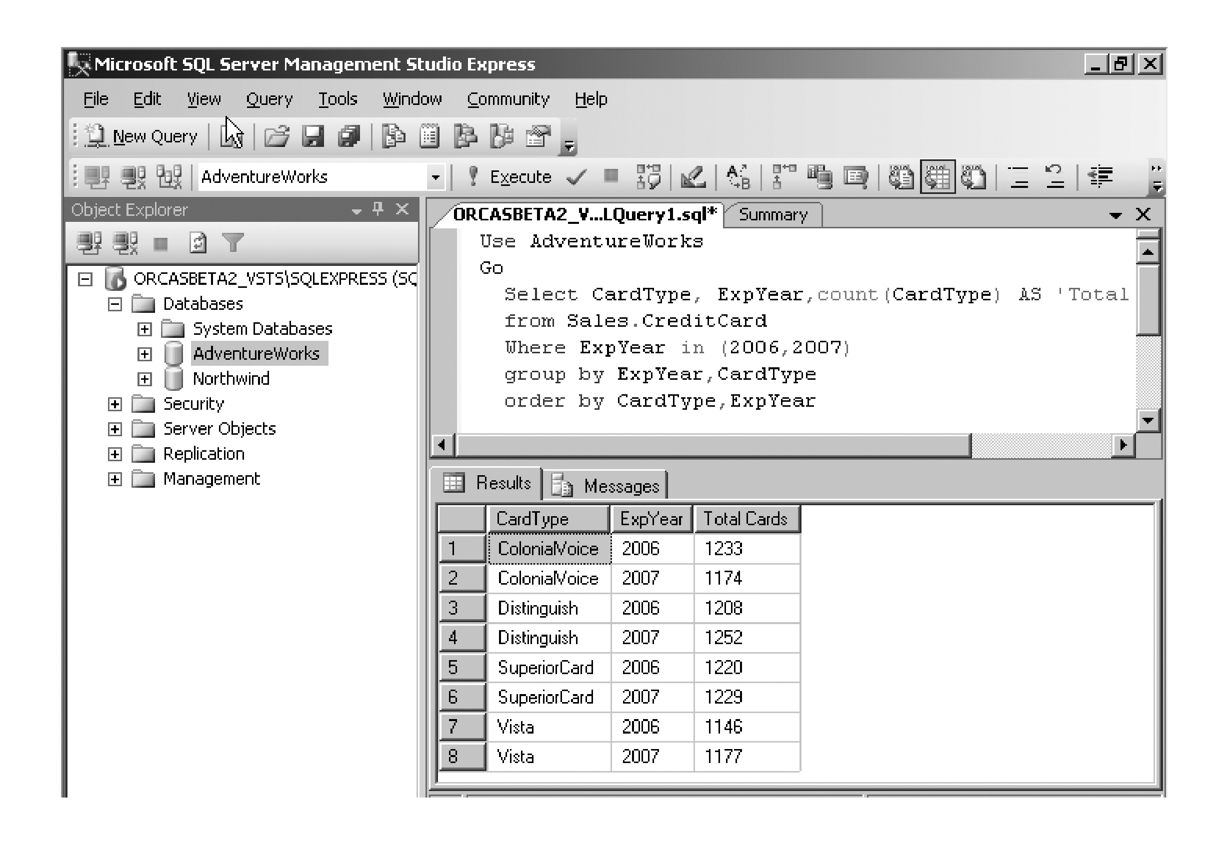
تستخدم هذه الجملة لتنظيم الصفوف Rows الناتجة من استعلام معين الى مجموعات ، واجراء جملة SELECT على بيانات فى صفوف معينة واعادة دمجها فى اعمدة ، كمثال على ذلك نفترض وجود جدول باسم مجموعة من الشركات وقيمة التعامل معها ، وانت تريد تقرير بإجمالى القيمة لكل شركة على حدة ، جملة GROUP BY هى ما تبحث عنه لذلك.

مثال على ذلك

افتح SSMSE واكتب هذا الإجراء

```
Use AdventureWorks
Go
Select CardType, ExpYear,count(CardType) AS 'Total Cards'
from Sales.CreditCard
Where ExpYear in (2006,2007)
group by ExpYear,CardType
order by CardType,ExpYear
```

ثم قم بعمل Execute ولاحظ الناتج كما بالشكل:



	CardType	ExpYear	Total Cards
1	ColonialVoice	2006	1233
2	ColonialVoice	2007	1174
3	Distinguish	2006	1208
4	Distinguish	2007	1252
5	SuperiorCard	2006	1220
6	SuperiorCard	2007	1229
7	Vista	2006	1146
8	Vista	2007	1177

ماحدث هو اجراء استعلام على الجدول CreditCard من قاعدة البيانات Sales لتعود فى صورة ثلاثة أعمدة مع الدالة Count التى ستعود بعدد أنواع الكروت فى الجدول

```
Select CardType, ExpYear,count(CardType) AS 'Total Cards'
from Sales.CreditCard
```

ثم قمنا بعمل حصر للكروت التى ستنتهى فى الفترة من 2006 الى 2007 عن طريق الشرط WHERE ، ثم تأتى جملة GROUP BY لتعرض العمودان CardType و ExpYear معا :

group by ExpYear,CardType

أما جملة ORDER BY فهى تضمن ترتيب الناتج اعتمادا على شرط معين وفى حالتنا هنا الترتيب سيكون ابجديا على العمود CardType كاعتبار اول وعلى ExpYear كاعتبار ثانى.

المعامل PIVOT

تظهر ميزة استخدام المعامل PIVOT فى حالة عمل جدولة لتقارير فى صورة ملخصات للبيانات ، مع انه جرت شهرتها فى عملية تحويل بيانات الصفوف الى اعمدة ، على سبيل المثال لو أننا أجرينا استعلاماً على الجدول Sales.CreditCard فى قاعدة البيانات AdventureWorks لحصر عدد كروت الائتمان من نوع معين والتى ستنتهي فى تاريخ معين.
فمثلا فى المثال الخاص بالجملة GROUP BY لاحظنا تكرار لصفوف تحوي التاريخين ٢٠٠٦ و ٢٠٠٧ باستخدامك المعامل PIVOT يُختصر هذا الأمر الى صورة أكثر فهما وتنظيماً.

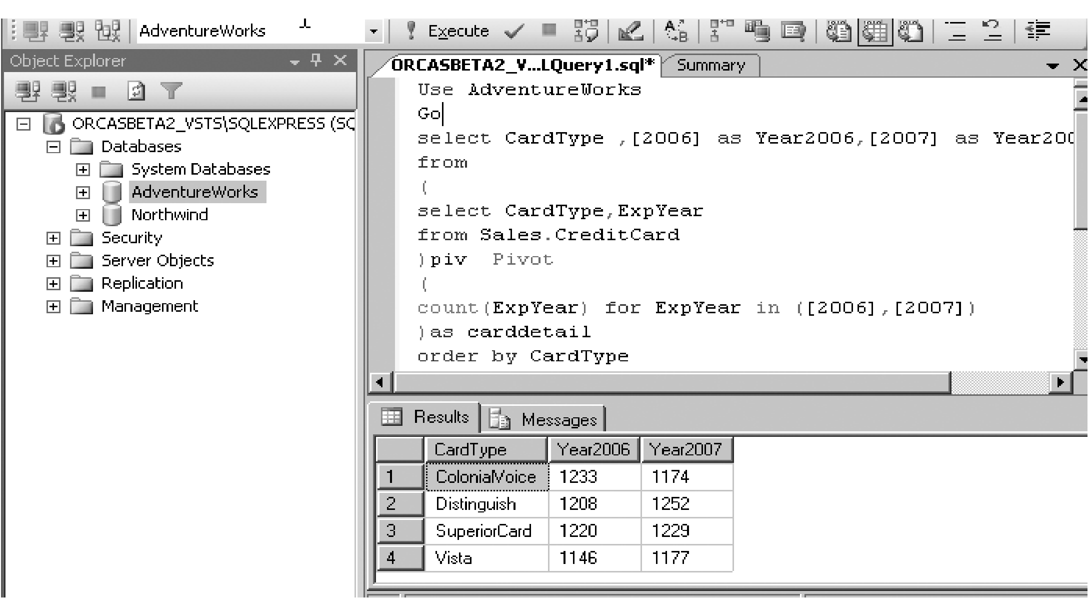
مثال على ذلك

ماذكرناه أعلى ، نريد أن نجري استعلاماً على الجدول Sales.CreditCard فى قاعدة البيانات AdventureWorks لحصر عدد كروت الائتمان من نوع معين تنهي فى تاريخ معين ، قم يفتح نافذة استعلام جديدة من الأمر New Query أدخل جملة الاستعلام التالية ثم اضغط Execute.

```
Use AdventureWorks
Go
select CardType ,[2006] as Year2006,[2007] as Year2007
from
(
select CardType,ExpYear
from Sales.CreditCard
)piv Pivot
(
count(ExpYear) for ExpYear in ([2006],[2007])
)as carddetail

order by CardType
```

سوف تلاحظ نتيجة كما بالشكل التالي:



لشرح ماحدث ببساطة أجرينا استعلاماً مستخدمين جملة SELECT مع الأخذ فى الاعتبار تغيير المسمي الحقيقي للأعمدة الى year2006 ,year2007 ثم اجرينا نفس الاستعلام مع انواع كروت الائتمان وتاريخ نهاية الصلاحية ثم ختمنا الجملة بالمعامل PIVOT ،ثم استخدمنا الدالة COUNT والتى تحصى عدد الكروت المنتهية فى العامين ٢٠٠٦-٢٠٠٧ ثم ختاماً ستخدمنا جملة ORDER BY والتى رتبت بناءاً على الدليل التالى لها وهو هذه المرة CardType .

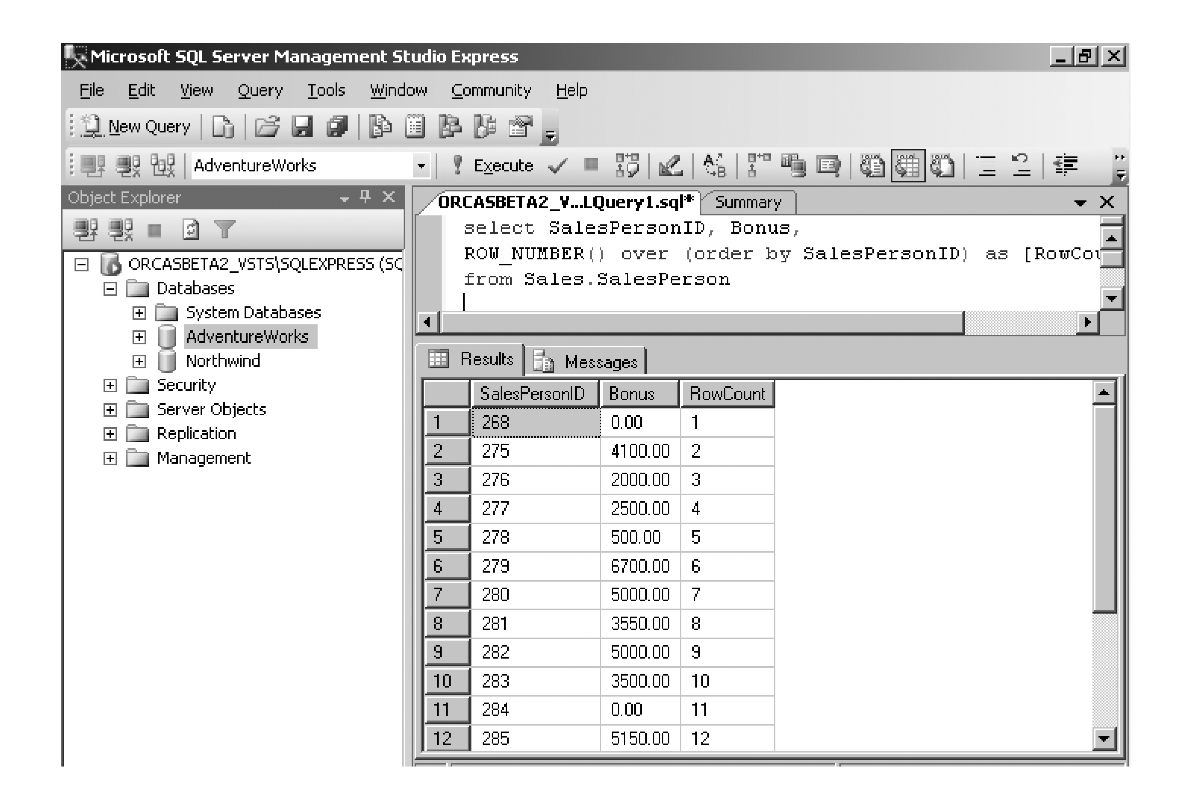
الدالة ROW_NUMBER

هى دالة تقوم بحصر عدد الصفوف فى جدول معين بحيث تعود برقم صحيح فريد غير قابل للتكرار.

مثال على ذلك

افتح نافذة استعلام جديدة فى SSMSE أكتب الإستعلام التالي واضغط Execute او F5 ولاحظ النتيجة :

```
select SalesPersonID, Bonus,
ROW_NUMBER() over (order by SalesPersonID) as [RowCount]
from Sales.SalesPerson
```



ماحدث أن استخدمنا هذه الجملة داخل جملة SELECT

ROW_NUMBER() over (order by SalesPersonID) as [RowCount]

وهى ببساطة تعنى (قم بعد عدد الصفوف للعائد من الاستعلام السابق مرتباً طبقا لـ SalesPersonID مع وضع الناتيج فى العمود RowCount)، مع ملاحظة هامة جدا أننا وضعنا RowCount بين قوسين مربعين [ ] وذلك كون هذه الجملة من الجمل المستخدمة فى SQL Server اى انها Key Word لكي تتأكد حاول استخدامها من دون أقواس .

جملة PARTITION BY

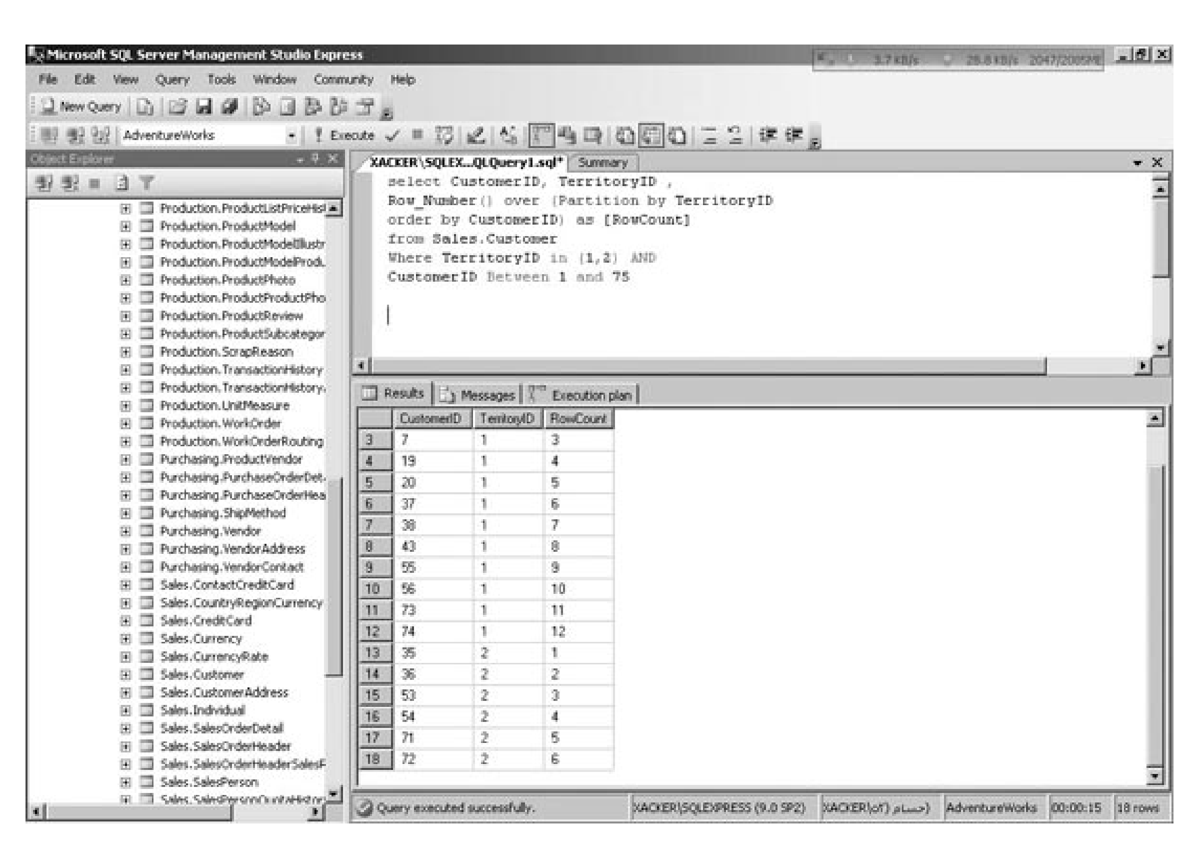
تستخدم هذه الجملة بالتوازى مع الدالة ROW_NUMBER( ) حيث يتم تقسيم الاستعلام الناتج من الدالة السابقة الى اجزاء Partitions لتعود بترتيب متسلسل لكل نتيجة مجزئة على حدا.

بالطبع نحتاج مثال

افتح نافذة استعلام جديدة ثم ادخل الكود التالى واضغط F5

```
select CustomerID, TerritoryID ,
Row_Number() over (Partition by TerritoryID
order by CustomerID) as [RowCount]
from Sales.Customer
Where TerritoryID in (1,2) AND
CustomerID Between 1 and 75
```

لاحظ أن العمود RowCount رتب أرقامه من ١ الى ...الخ وهذا الترتيب يكرر فى المقابل مع كل تغيير فى قيمة العمود TerritoryID



كما بالشكل حدث انقلاب للعد فى العمود RowCount فى مقابل التغيير الحادث للعمود TerritoryID.

لشرح ما تم فإن الجملة :

```
Row_Number() over (Partition by TerritoryID
order by CustomerID) as [RowCount]
```

والتى تم استخدامها كجزء من الجملة SELECT تستعمل بالتجانس فيما بين الجمل Row_Number() , over and Partition by لتقوم بهذا التقسيم بناءاً على شرط وليكن كما فى مثالنا Where TerritoryID in (1, 2).

توافق القوالب Pattern Matching

وفيها يتم اختبار حروف من سلسلة نصية ما ان كانت تنتمى الى pattern معينة أم لا. ولتحديد pattern معينة تستخدم الحروف العادية وكذا الرموز الاستعاضية مثل (% أو _). لابد وأن تتوافق حروف السلسلة النصية المراد مقارنتها مع الـ pattern المحددة مسبقا . تستخدم الجمل LIKE , NOT LIKE فى اختبار توافق النص مع القالب المحدد مع ملاحظة ان القوالب حساسة لحالة الأحرف Case Sensitive. يدعم Sql Server هذه الرموز الاستعاضية بديلا عن الأحرف فى السلاسل النصية المراد مقارنتها.

- (% النسبة المئوية) وهى بديل عن صفر الى أكثر من واحد من الحروف ، مثال ايجاد كل العناوين التى تحتوى على C# 2010 بغض النظر عن كيفية وقوع هذه السلسلة من الجملة فى البداية –المنتصف –النهاية لا يهم :

WHERE title LIKE '%C# 2010%'

تكون النتيجة الرجوع مثلا بهذه العناوين :

“C# 2010: An Introduction,” - “Accelerated C# 2010 “ – "Beginning C# 2010 Databases"

- (_ underscore) وهى بديل عن حرف واحد فقط وباستخدامها تكون محدد فى اختيارك للنتائج المرجوة ، مثال على ذلك الجملة التالية تعود باسم المؤلف فى قاعدة بيانات والذى يتكون من اربعة حروف تنتهى بالحروف الثلاثة ean :

SELECT * FROM books_db WHERE au_fname LIKE '_ean'

لتعود لنا النتيجة بالأسماء التالية مثلاً (Sean –Kean-Dean ….etc) ولو تحايلنا على الاستعلام بوضع أكثر من _ لتعود بأكثر من نتيجة مثلا WHERE au_fname LIKE 'a___n' ومعناها ابحث عن اسم المؤلف الذى يبدأ بـ (a) وينتهى بـ (n) ومكون من خمسة أحرف لتعود باسماء على سبيل المثال :

Allan –Amman

- ([ ] square brackets) وفيها يتم تحديد مجموعة أحرف بديلة عن المراد البحث بها فمثلا [a-f] تعنى استعاضة الاحرف من a الى f مكان النص المراد البحث عنه وكذا تكون صحية أيضا لو حددنا الحروف يدويا مثل [abcdef]

مثال :

WHERE au_fname like '[C-K] arsen'

وتعنى البحث عن اسم المؤلف الذى ينتهى بـ arsen ويبدأ بحرف ابتداءاً من C الى K مثل Carsen, Darsen, Larsen, Karsen.

<u>ملاحظة قيمة :</u>
<u>يمكنك استخدام الرمز ^ قبل أى من المجموعات المحددة وذلك لعدم البحث فى هذا النطاق مثل [^a-f] أى لا تستبدل بأى من الحروف الموجودة داخل الأقواس</u>

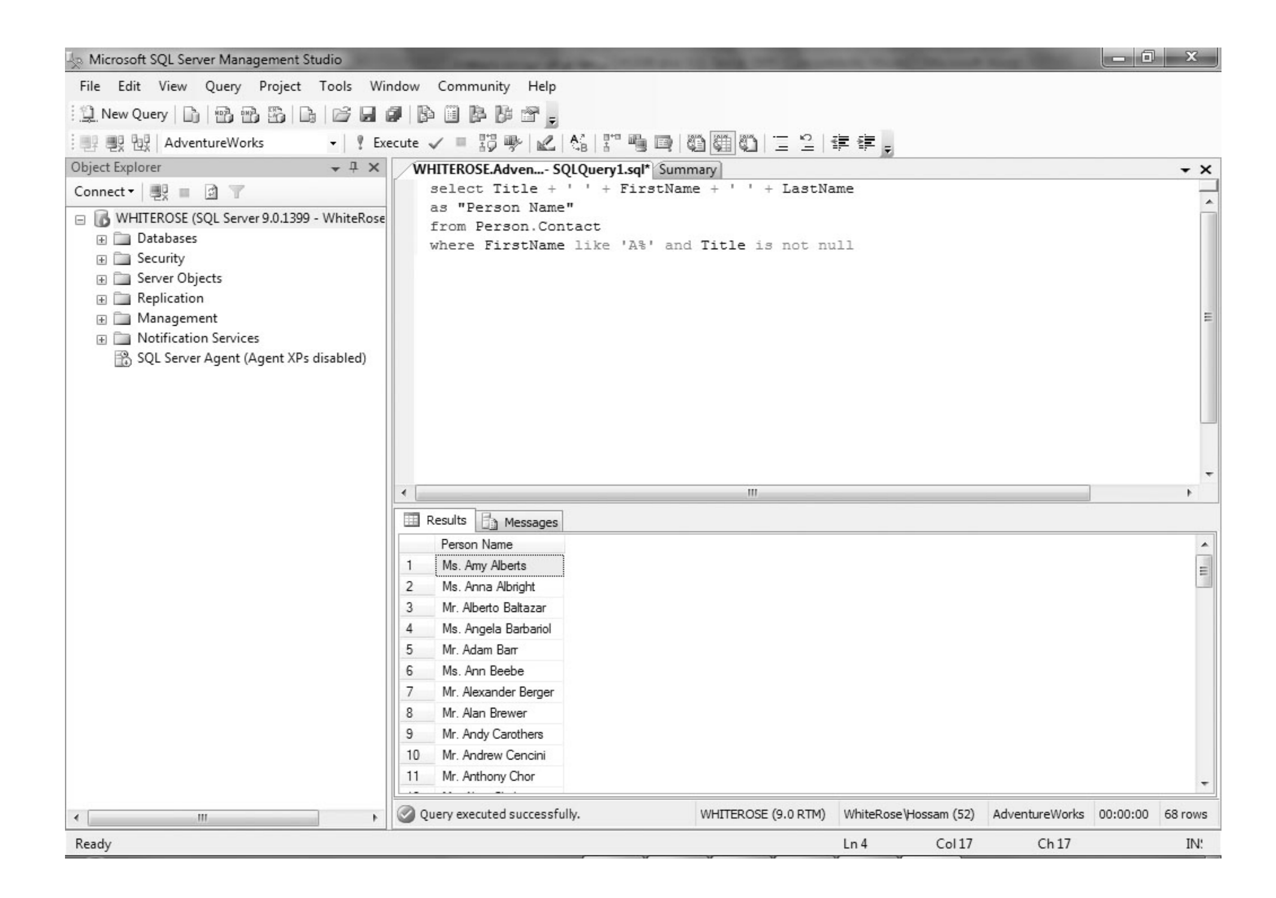
أمثلة جربها بنفسك

أولا استخدام العلامة المئوية % :

افتح SSMSE افتح نافذة استعلام جديدة كما تعلمت فى السابق ، اكتب هذا الاستعلام :

```
select Title + ' ' + FirstName + ' ' + LastName
as "Person Name"
from Person.Contact
where FirstName like 'A%' and Title is not null
```

لتكون النتيجة كما بالشكل :



ماحدث أن قمنا بتجميع الثلاثة أعمدة Title –FirstName –LastName في عمود واحد تحت مسمى Person Name باستخدام الجملة :

select Title + ' ' + FirstName + ' ' + LastName
as "Person Name"
from Person.Contact

ثم قمنا بعمل شرط مستخدمين الجملة التالية :

WHERE FirstName like 'A%' and Title is not null

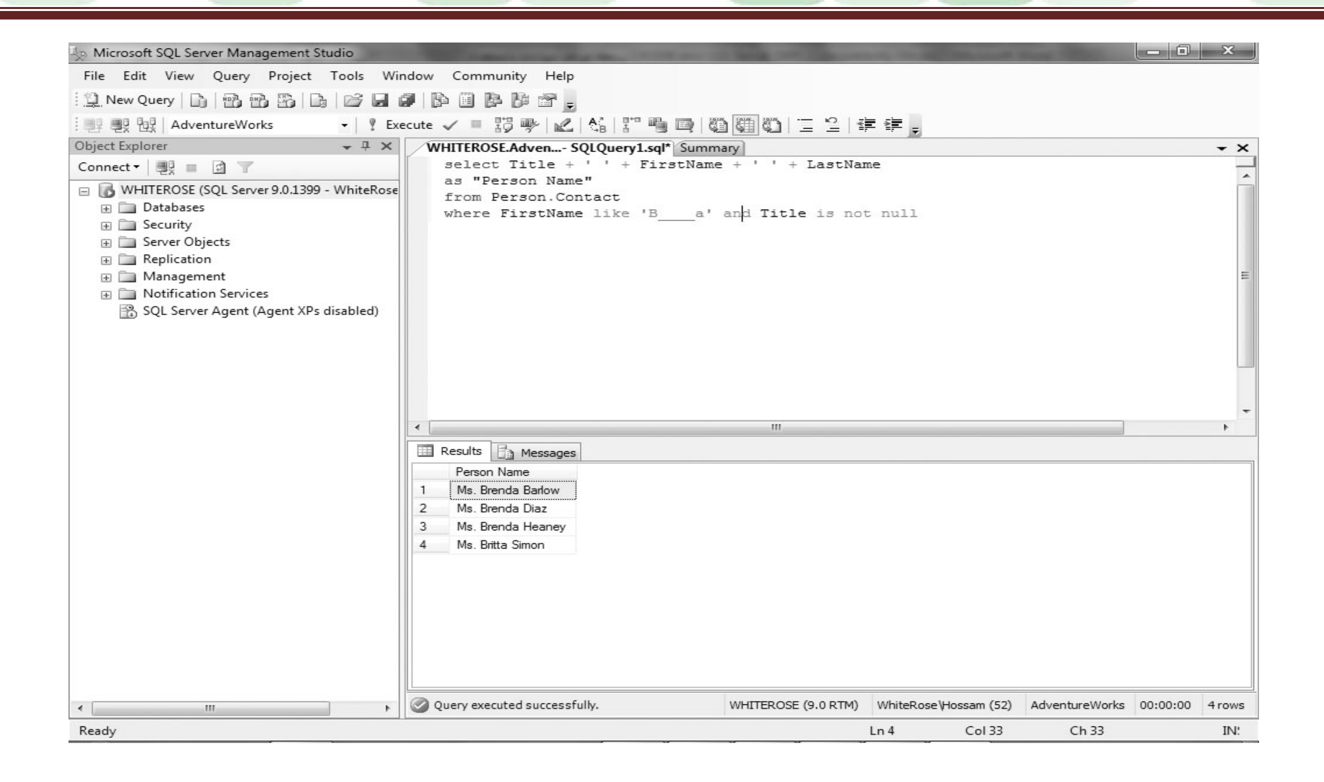
وتعنى ان يكون الاسم الأول يبدأ بالحرف A لأى عدد من الحروف يعقبه

مثال باستخدام _ :

اكتب الجملة التالية فى نافذة استعلام جديدة :

select Title + ' ' + FirstName + ' ' + LastName
as "Person Name"
from Person.Contact
where FirstName like 'B____a' and Title is not null

لتكون النتيجة كما بالشكل:



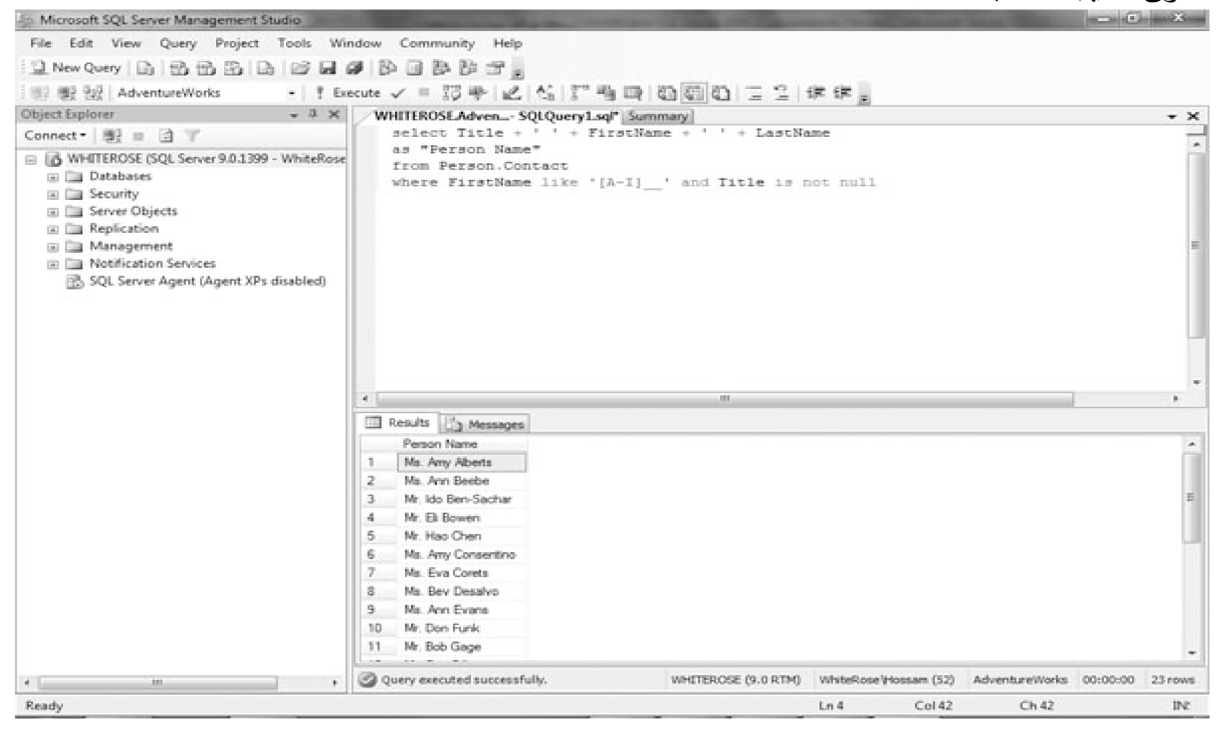
والشرح كما بالسابق مع اختلاف ان (_) تستعيض عن حرف واحد فقط .

مثال باستخدام [ ] (Square Bracket):

افتح نافذة استعلام جديدة واكتب الجملة التالية :

```
select Title + ' ' + FirstName + ' ' + LastName
as "Person Name"
from Person.Contact
where FirstName like '[A-I]__' and Title is not null
```

لتكون النتيجة كما بالشكل:



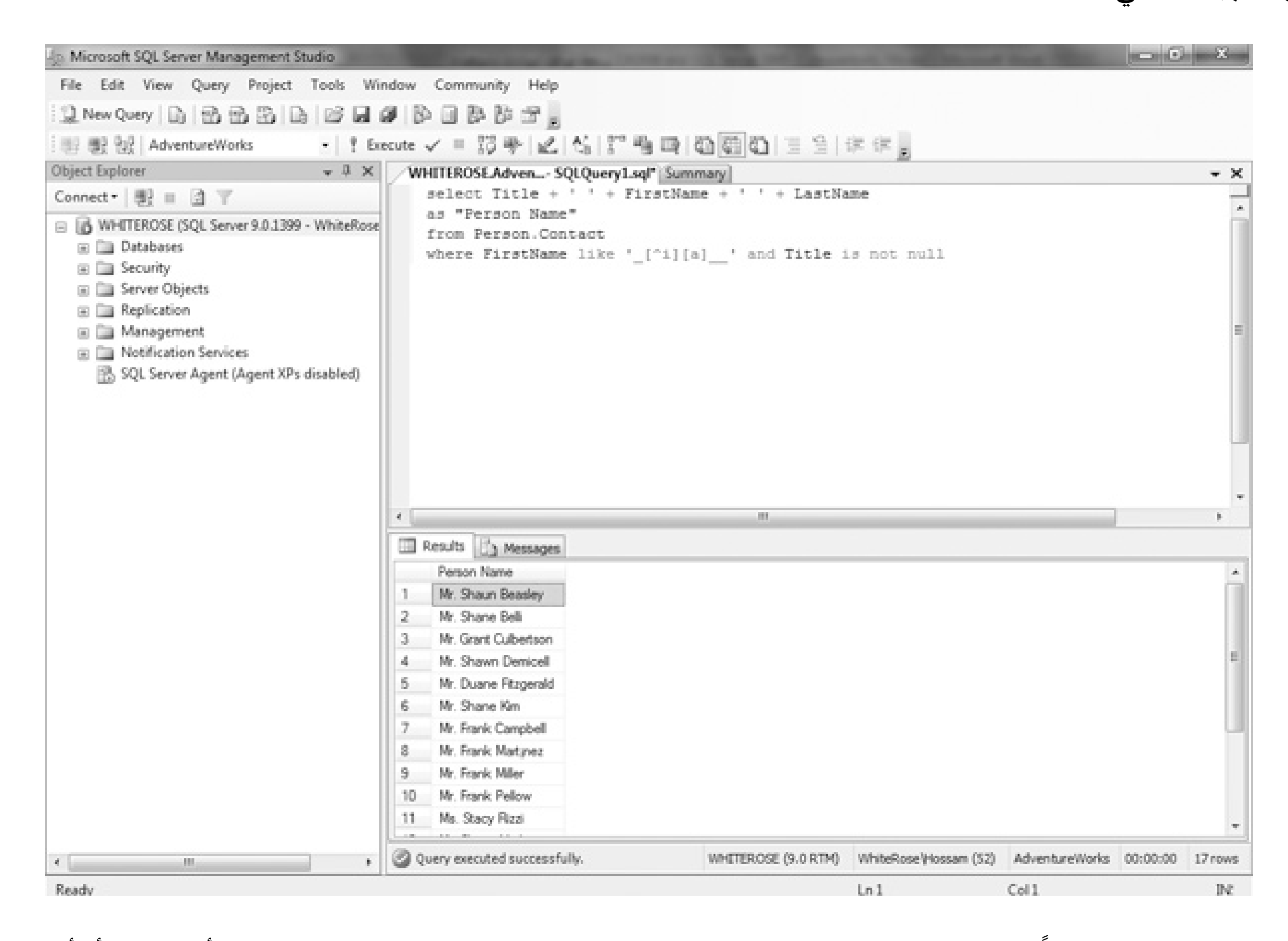
ماحدث كما بالسابق مع اختلاف الاستعاضة هذه المرة فالبحث عن الاسم الاول المكون من ثلاثة أحرف والذى يبتدئ بحرف ضمن مجموعة الحروف من A الى I وينتهى بأى حرفين.

مثال باستخدام [^] (Square Bracket and Caret):

افتح نافذة استعلام جديدة واكتب الجملة التالية :

```
select Title + ' ' + FirstName + ' ' + LastName
as "Person Name"
from Person.Contact
where FirstName like '_[^i][a]__' and Title is not null
```

لتكون النتيجة كالتالي:



ولفهم ماحدث كالسابق تماماً باختلاف جملة المقارنة والتى تعنى ان يكون الاسم الاول مكون من خمسة أحرف تبدأ بأى حرف والحرف التالى لايكون i اما الحرف الثالث فيكون a متبوعاً بأى حرفين أخرين.

دوال التجميع Aggregate Functions

تحتوى لغة SQL على دوال ضمنية لعمل حصر او تجميع لمحتوى الأعمدة من قيم.وتُطبق على العديد من الصفوف لتعود بقيمة واحدة.على سبيل المثال يًمكنك استخدام أحد هذه الدوال فى حساب متوسط سعر سلعة ما فى عمود.وكذا يُمكنك حساب أقل أو أكبر سعر بالطبع.من دوا التجميع التى سنتعامل معها AVG,MAX,MIN,SUM,COUNT.

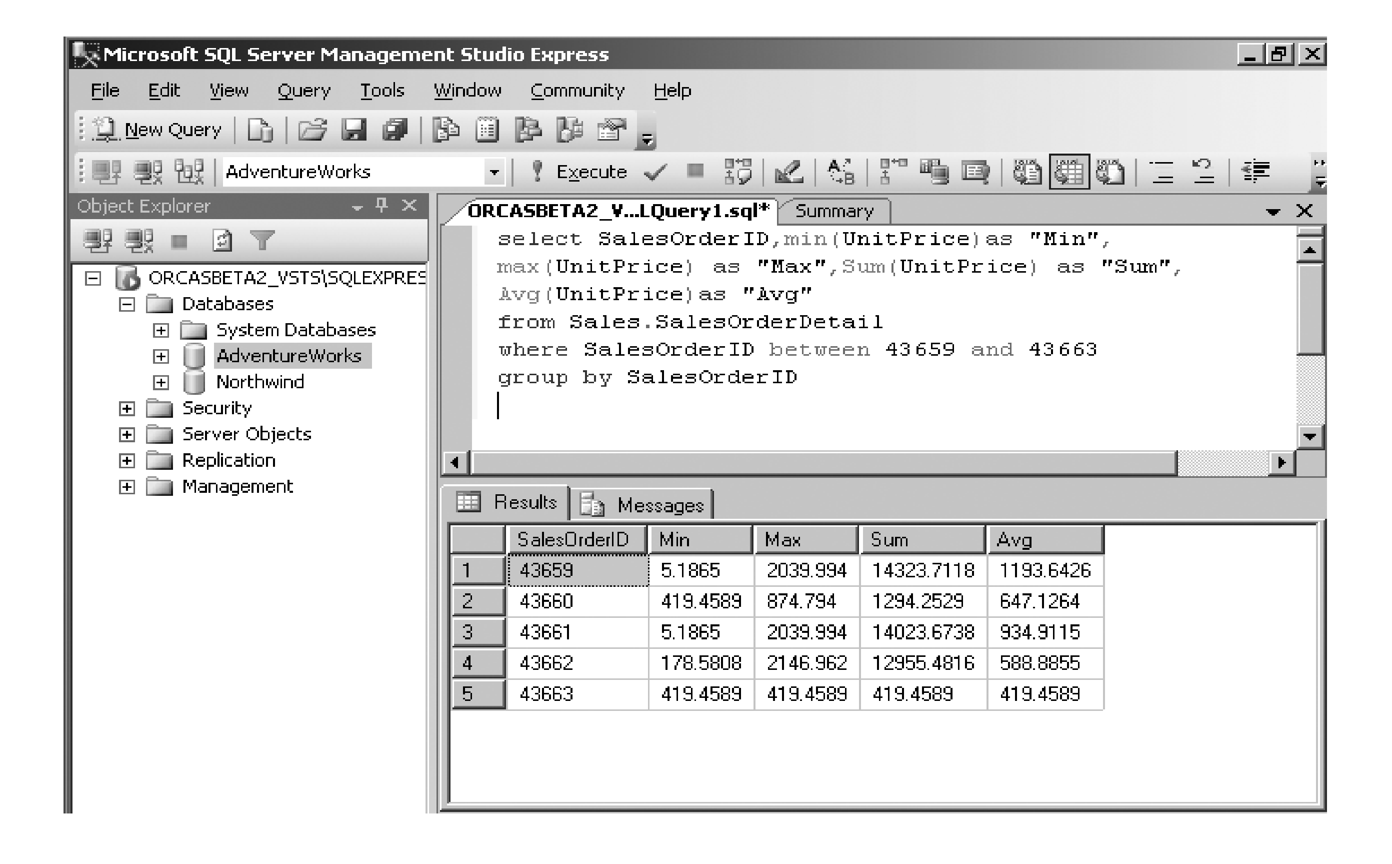
مثال على ذلك

تطبيق ماذكرناه بأعلى كمثال على حساب متوسط وأقل وأكبر ومجموع سعر وحدة (UnintPrice) لكل أمر بيع (SalesOrderID) من الجدول SalesOrderDetail.

افتح نافذة استعلام جديد وادخل هذا الكود واضغط Execute او F5 للتنفيذ :

```
select SalesOrderID,min(UnitPrice)as "Min",
max(UnitPrice) as "Max",Sum(UnitPrice) as "Sum",
Avg(UnitPrice)as "Avg"
from Sales.SalesOrderDetail
where SalesOrderID between 43659 and 43663
group by SalesOrderID
```

لتشاهد كما بالشكل:-



لشرح ماحدث فإننا استخدمنا الدوال MIN و MAX وذلك لحساب أصغر وأكبر قيمة للسعر وكذا استخدام الدالتين SUM و AVG لحساب المجموع ومتوسط القيمة على الترتيب

```
min(UnitPrice) as "Min",
max(UnitPrice) as "Max",
Sum(UnitPrice) as "Sum",
Avg(UnitPrice)as "Avg"
```

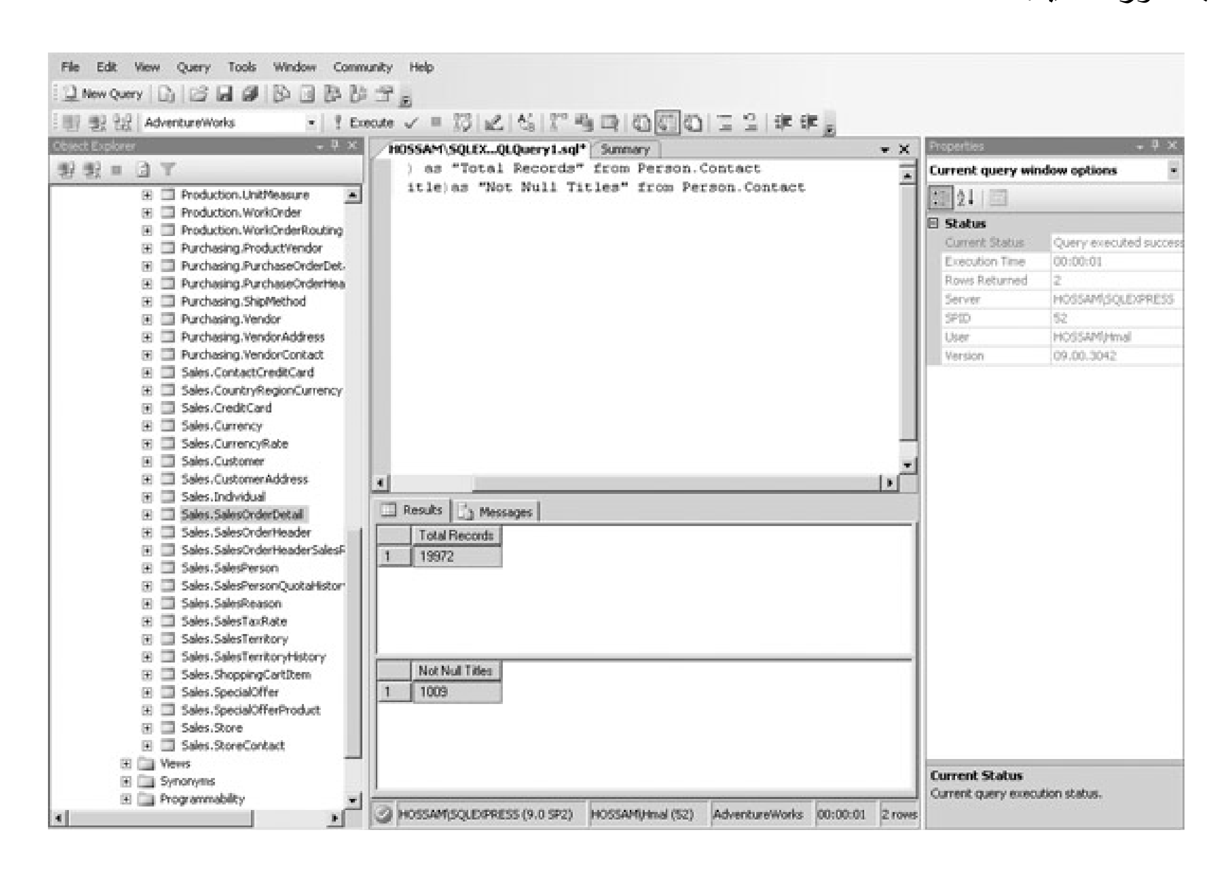
وقمنا باستخدام الدالة GROUP BY لترتيب العناصر حسب SalesOrderID. من النتائج المعروضة نري أن أقل قيمة للسعر بالنسبة لـ Order 1 هى ٥,١٨٦٥ وأن أقصى قيمة هى ٢٠٣٩,٩٩٤ وان متوسط السعر هو ١١٩٣,٦٤٢٦ وإجمالى سعر الوحدات هو ١٤٣٢٣,٧١١٨.

جرب الدالة COUNT :

لنقم بعد السجلات الخاصة بالجدول Person.Contact.
افتح نافذة استعلام جديدة كما هو معتاد وأدخل الكود التالى ثم F5:

```
Select count(*) as "Total Records" from Person.Contact
Select count(Title)as "Not Null Titles" from Person.Contact
```

لتشاهد كما بالصورة التالية:



لفهم ماحدث فإن الدالة COUNT لها استخدامات عديدة تتوقف على البرامتر Parameter المضاف اليها ، فمثلا COUNT(*) تعنى حصر جميع السجلات الممكنة كما بالمثال أعلى وهى ١٩٩٧٢ سجل من الجدول Person.Contact.
أما لو استخدمت اسم العمود كبرامتر فإن القيم التى ستعود هى القيم التى لاتحوي NULL ففى المثال قمنا بوضع العمود Title كبارامتر للدالة COUNT ومع أن عدد السجلات الحقيقى فى المثال هو ١٩٩٧٢ سجل إلا أن ناتج هذه العملية هو ١٠٠٩ سجل فقط ! وهو مايعنى تجاهل السجلات ذات القيمة NULL.

دوال الوقت والتاريخ DATETIME Functions

على الرغم من أن النوع DATETIME معرف ضمنيا فى لغة SQL القياسية مشتملاً أيضاً على مكوناته YEAR,MONTH,DAY,HOUR,MINUTE,SECOND ، إلا أنها لاتوضح كيفية تعامل نظام إدارة قواعد البيانات (DBMS) معه فيما يعنى أن كل DBMS له طريقته فى التعامل مع هذا النوع وهذه الدوال.

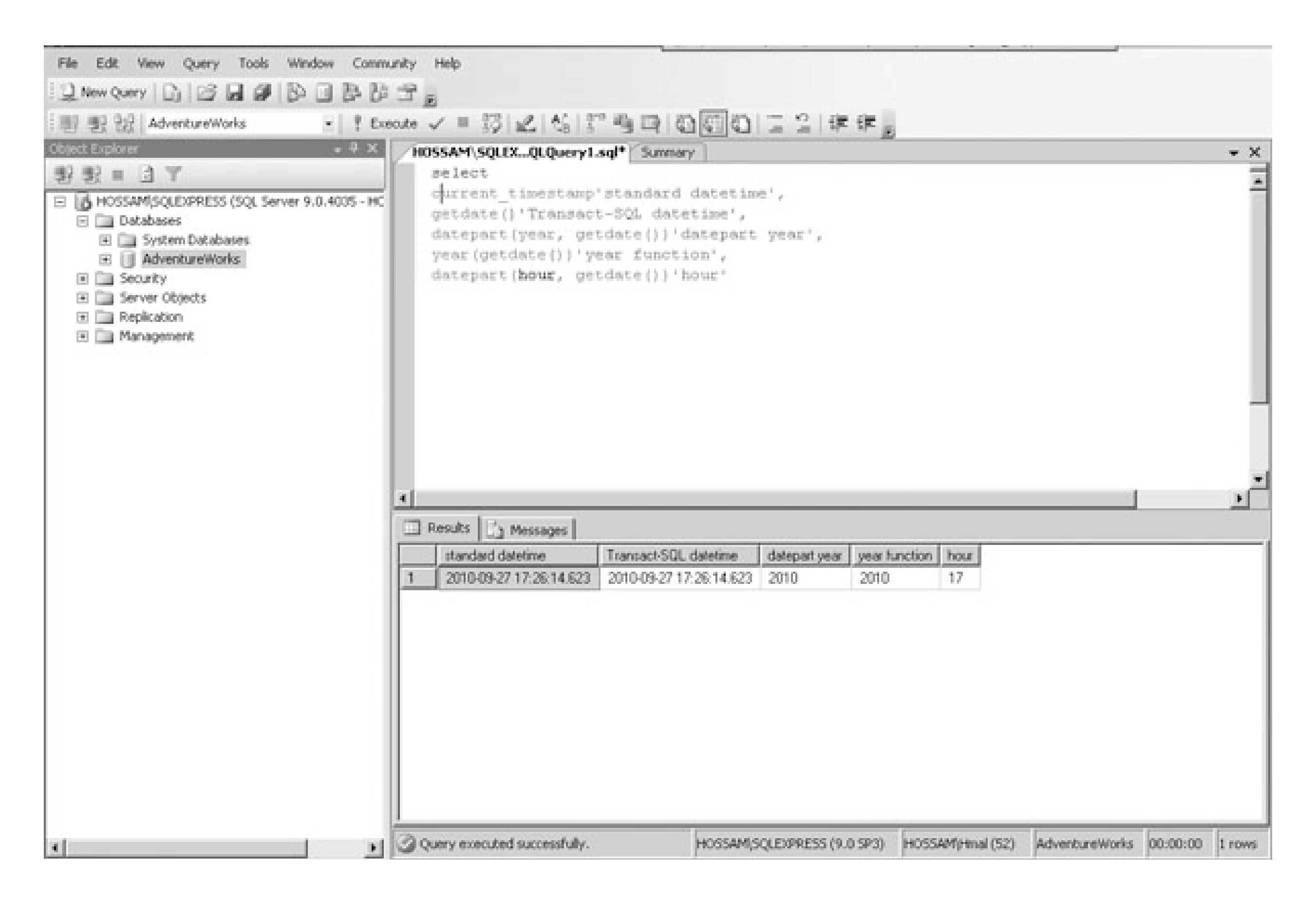
جرب التعامل مع دوال الوقت والتاريخ فى T-SQL

افتح نافذة استعلام جديد وأدخل هذا الكود ثم اضغط Execute :

```
select
current_timestamp'standard datetime',
```

getdate()'Transact-SQL datetime',
datepart(year, getdate())'datepart year',
year(getdate())'year function',
datepart(hour, getdate())'hour'

لتشاهد كما بالصورة



تعودنا حينما نرى جملة SELECT أن نلحقها بـ FROM لكن هذا فى الصيغة القياسية ، أما فى مثالنا هذا فإن أول سطرين من الاستعلام مسئولان عن جلب التاريخ والوقت

select
current_timestamp'standard datetime',
getdate()'Transact-SQL datetime',

السطر الأول يستخدم الجملة CURRENT_TIMESTAMP وهى جزء من لغة SQL القياسية ، اما الدالة GETDATE() فهى جزة من لغة T-SQL فكلاهما يعطى نفس الناتج.
فى السطرين التاليين

datepart(year, getdate()) 'datepart year',
year(getdate()) 'year function',

فإن الدالة DATEPART تعمل على تقسيم التاريخ على حسب المعامل المضاف لها مثل حالتنا هذه YEAR ، أما الدالة YEAR فهى تجلب السنة مباشرة من تاريخ معين وفى حالتنا هنا حصلنا على معامل التاريخ باستخدام الدالة GETDATE().

فى السطر الأخير حصلنا على الساعة الحالية (لاحظ ان T-SQL لايوفر دالة مستقلة للساعة مثل الدالة YEAR) فقمنا باستخدام الدالة مقسمة التاريخ GETDATE() مع المعامل HOUR للحصول على الساعة

datepart(hour, getdate())'hour'

تستطيع تنسيق واستجلاب الوقت والتاريخ والعديد من المزايا الأخرى بعدة طرق ، فيمكنك زيادة الوقت والانقاص منه أياً ما كان نوع DBMS الذى تعمل عليه ، لكن انتبه ! لأنك ستجد أصعب نوع بيانات ستعمل عليه هو الوقت والتاريخ وستضطر لتتعامل بحرفيه مع هذه الدوال البسيطة المدرجة ضمن T-SQL او ماشابهه من لغات.

تنويه

تأكد عندما تستخدم بياناتمن نوع DATETIME ان تدخله بطريقة معينة مثل 3/21/2003 فهذه هى أنسب طريقة لإخبار DBMS ان هذا السجل يحوى بيانات من النوع DATETIME، لكن قبل هذا عليك قراءة تعليمات SQL الخاصة بالنظام الذى تعمل عليه.

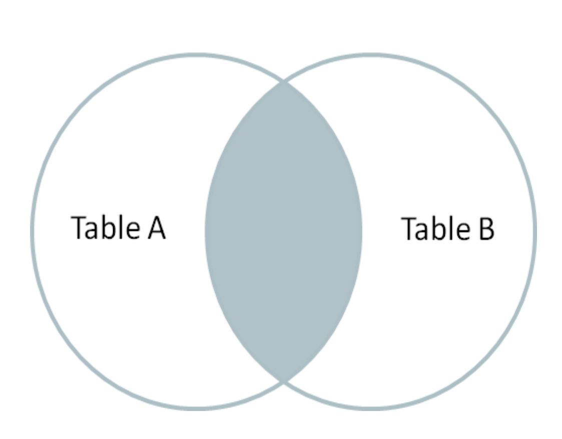
فهم JOINS

فى أغلب الأحيان قد تضر لعمل استعلام من أكثر من جدول ، ولحل هذه المشكلة فإنك حتماً ستضطر لاستخدام جملة JOIN.
لفهم ماهية JOIN فإن دورها يقوم على الاستعلام عن بيانات من جدولين أو أكثر (ولا يلتزم التمايز) ومطابقة الحقول بكل منهم بناءً على خيارات أنت تحددها.
هناك انواع كثيرة من joins والتى تتحد فى فكرة أنها ذات عمليات ثنائية حتى ولو كانت على نفس الجدول (كما سترى).
عمليات الـ joins مزعجة وأحياناً تبدو معقدة بعض الشئ لكن لاتقلق سأحاول تبسيط الأمر بما يسر الله لى.
سنستخدم فى عمليات الـ joins قاعدة البيانات الشهيرة Northwind بالطبع انت تعلم كيف تحصل عليها كما سترى فى الملحق.

Inner Joins

وهى أشهر أنواع الـ joins اذ تقوم بإعادة صف واحد طبقاً لما حددته فى استعلامك وغالباً المشترك بين جدولين كما فى الشكل ، على الرغم من ان اى عامل ارتباط Relational Operator يصلح نظرياً فى خصائص جملة join مثل (< أو >) إلا أن العامل (=) هو الأكثر استخداما وهذا النوع من joins يُسمى natural joins.
تتكون جملة join من التالى:-

```
select
<select list>
from
left-table INNER JOIN right-table
ON
<join specification>
```



سالفاً اخبرتكم أن join عملية ثنائية Binary Operation فيما يعنى كما تعلمنا من الرياضيات أنها تحتاج إلى طرف أيمن وطرف أيسر ولهذا نرى الجملة left-table INNER JOIN right-table ولا يعنى هذا لزاماً ان يكون أحد الطرفين جدول خالص ، إنما قد يكون اى شئ مستخلص من استعلام ما او حتى جدول ناتج من عملية join اخرى.
الكلمة ON هى التى تحدد متطلبات جملة join يعنى باختصار أى شرط يتحقق بالجملة WHERE.

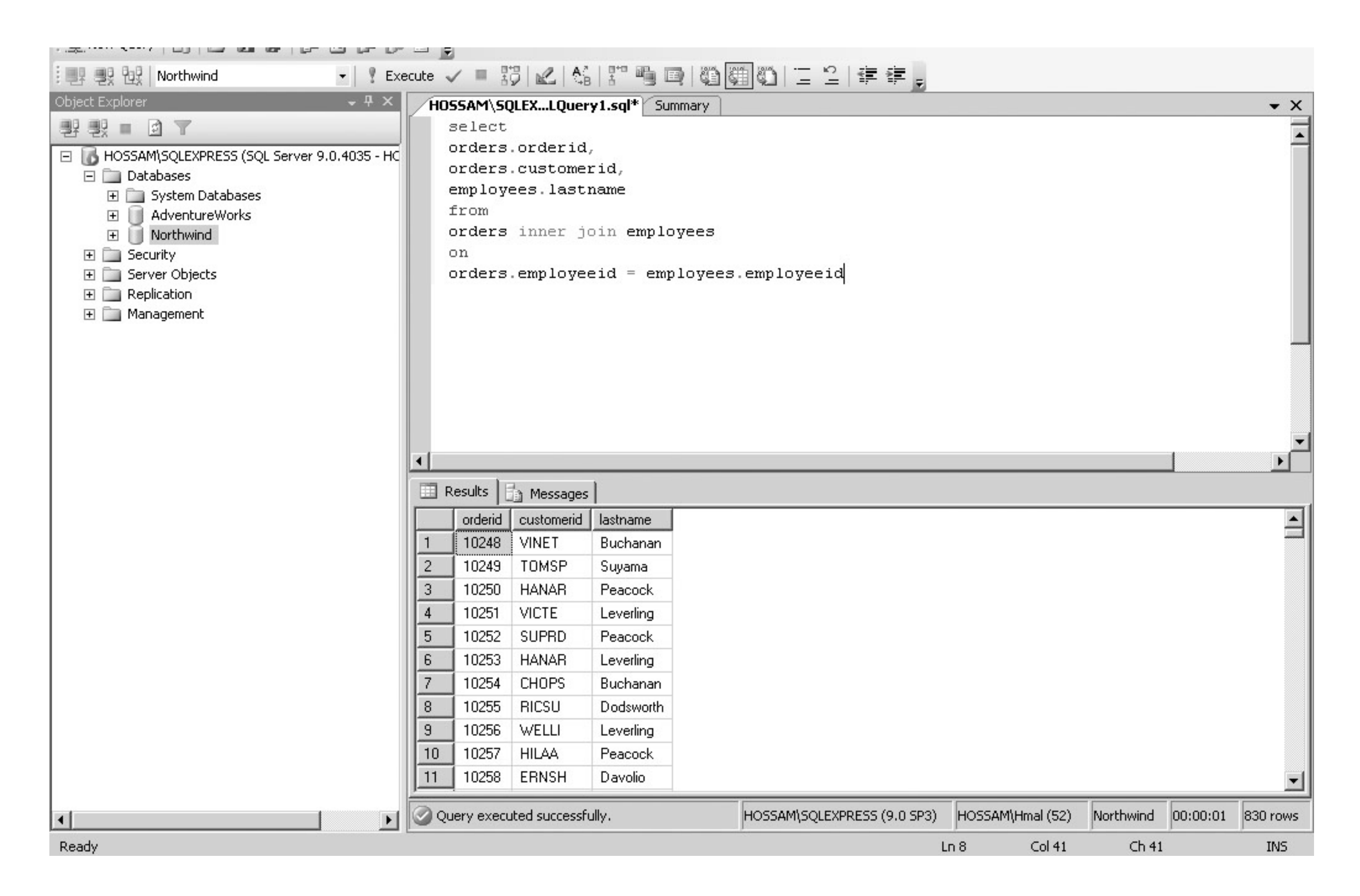
جرب هذا المثال لتقريب الصورة

سنقوم بطلب قائمة الطلبيات بمعنى رقم الـ ID للعميل الذى طلب ..و last name للموظفين employees القائمين على ذلك.

افتح نافذة استعلام جديدة (تذكر أننا نعمل على قاعدة البيانات Northwind) وأدخل هذا الاستعلام واضغط F5

```
select
orders.orderid,
orders.customerid,
employees.lastname
from
orders inner join employees
on
orders.employeeid = employees.employeeid
```

لتشاهد كما بالصورة التالية:



هل لاحظت معى إختلاف الاستعلام مع جملة SELECT هذه المرة ؟! ...بالتأكيد اختلف والسبب فى اننا نستعلم عن اكثر من عمود من جدولين مختلفين وبالتالى كان لابد من تمايز كل عمود بجدوله باستخدام الفاصلة (.) مع اسم الجدول يسبقها و هذا متعارف عليه بإسم إزالة الالتباس او *disambiguation* ولهذا لايجد نظام إدارة قواعد البيانات صعوبة أو مشكلة فى معرفة المطلوب منه ، هذا الاجراء أيضاً لا يلزم أن يكون كل عمود بجدول منفصل ..إنما يمكنك استخدامه فى أى استعلام عادى حتى ولو كان من نفس الجدول.

جملة FROM هنا استخدمت اسم الجدولين متوسطاً بنوع join والتى فى حالتنا هنا inner join

```
from
orders inner join employees
```

وهى توضح ضابط ربط المفتاح الأساسي P.K (EmployeeId) للجدول Employees بالمفتاح الخارجي Foreign Key (EmployeeId) للجدول Orders .

on
orders.employeeid = employees.employeeid

لاحظنا ان ناتج الاستعلام السابق مكون من ثلاثة أعمدة OredrID , CustomerID , LastName وهى ناتج الصفوف فى الجدولين التى يشترك فيها العمود EmployeeID نفس القيمة وبالتالى فإن اى قيمة فى أى صف لا تتوافر فيها هذه الشروط يتم استبعادها.

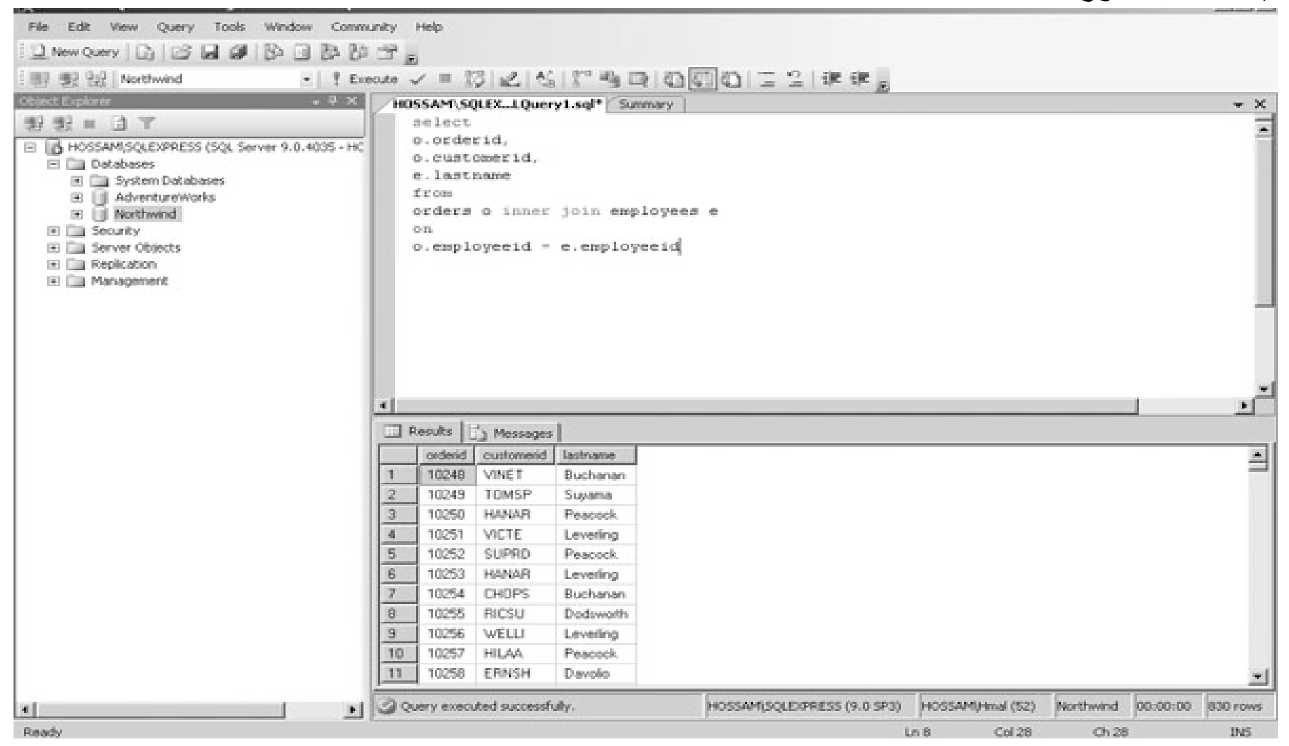
ملحوظة: هل انتبهت إلى ان EmployeeID لم يُدرج فى جملة SELECT مع أن مدار عمل join فى هذا المثال عليه ؟!!

نفس المثال لكن مع Correlation name

افتح نافذة استعلام جديدة (لازلنا على قاعدة Northwind) ادخل هذا الاستعلام ثم Execute

```
select
o.orderid,
o.customerid,
e.lastname
from
orders o inner join employees e
on
o.employeeid = e.employeeid
```

لترى النتيجة كما بالصورة



هل لاحظت أن النتيجة هى هى لم تتبدل ؟!

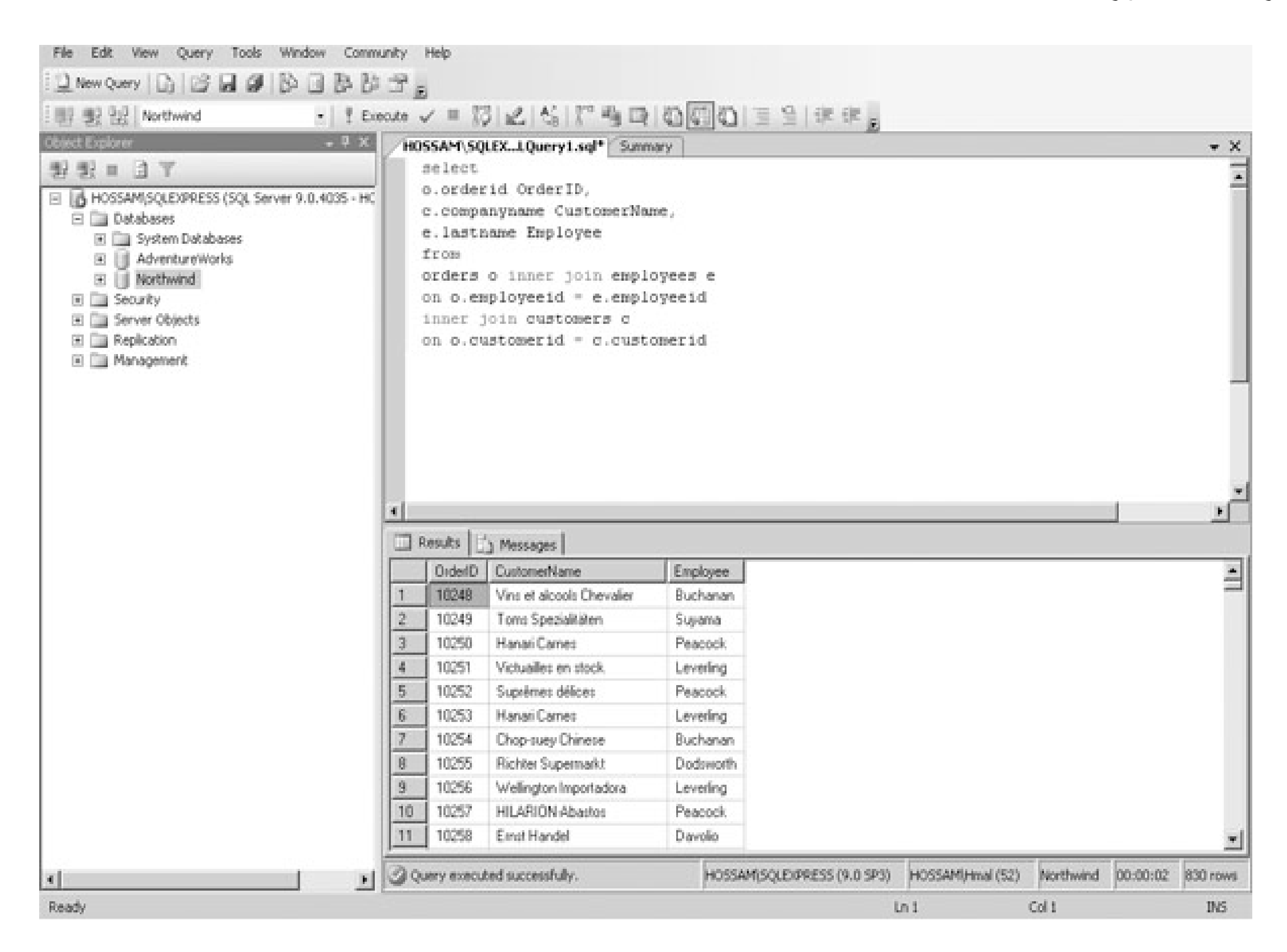
قمنا باستخدام ما يُسمى *Correlation name* لكل جدول ، وهى بمثابة اشارة مرجعية reference كما فى البرمجة العادية (بالطبع أسمعك تقول أن هذا ماهو إلا aliases لاسم العمود ، صحيح أن الطريقة مشابهه إلا انه ليس كذلك حيث ان هذه الطريقة تحل محل الجدول فعلياً وليس مجرد عنونة للإسم كما فى aliases)، فبإمكانك الآن أن تشير إلى Oreders بـ o وإلى Employees بـ e وهذا أسهل كثيراً أثناء كتابتك للإستعلام بدلاً من استخدام الإسم مطولاً.

مثال آخر مع ثلاث جداول

افتح نافذة استعلام جديدة (لازلنا على قاعدة Northwind) ادخل هذا الاستعلام ثم Execute

```
select
o.orderid OrderID,
c.companyname CustomerName,
e.lastname Employee
from
orders o inner join employees e
on o.employeeid = e.employeeid
inner join customers c
on o.customerid = c.customerid
```

لترى كما بالصورة التالية



اختلافاً عن المثال السابق قمنا باستبدال CustomerID من الجدول Orders بـ CompanyName من الجدول Customers فى جملة SELECT.

```
select
o.orderid OrderID,
c.companyname CustomerName,
e.lastname Employee
```

ثم قمنا بإضافة inner join ثانية لأننى كما وضحت سابقاً اخبرتك انها ثنائية الجهة مهما كانت طبيعة الاستعلام فأصبح لدينا جدول ناتج عن الـ join الأولى يتم ادخاله فى الـ join الثانية مع الجدول Customers.راع دائماً فى كتابتك لأى استعلام أن

يكون منسقاً كى يكون مقروءاً لهذا قسمنا الإستعلام إلى ثلاثة أسطر.يُمكنك أن تستخدم الأقواس فى حالة الاستعلام الكبير أو المشترك كما فى حالتنا هنا اكثر من inner join ، (بما أن الناتج جدول يُمكنك إدخاله فى correlation name لكن هذا الاستعلام سيكون متشابك وسيُربكك ☺).

```
from
orders o inner join employees e
on o.employeeid = e.employeeid
inner join customers c
on o.customerid = c.customerid
```

لاحظ ان عملية المقارنة ستتم بين الجداول الثلاثة لذا فإن كل الصفوف الخاصة بـ Orders تم عمل matching لها مع بالجدولين الآخرين.
من الطبيعى ان نتسائل عن كيفية تلبية نظام قواعد إدارة قواعد البيانات لهذه العمليات ،الأمر يكمن فى أن joins جزء لايتجزأ من عمليات قواعد البيانات العلائقية حيث تلجأ إلى تحسين optimize طريقة التعامل مع أكثر من جدول مع مراعاة انه كلما استخدمت joins أقل كلما ازدادت كفاءة استعلامك و هذا يعود لجودة تصميمك لقاعدة البيانات وخبرتك فى الاختصار على نفسك برمجياً.لكن ماذا عن الجداول التى لا تحوى صفوف متماثلة ؟! هذا هو التالى ..

Outer Joins

تعمل outer joins على إرجاع كل الصفوف من (على الأقل) واحدٍ من الجداول المشتركة فى هذه العملية حتى ولو كانت هذه الجداول غير متماثلة doesn't match.هناك ثلاثة أنواع من outer joins اولها : left outer join وثانيها : right outer join وثالثها : full outer join . إن left, right ماهى إلا إشارة إلى جهة الـ operand من معامل الـ join .
النوع الأول left outer join فإن كل صفوف الجدول الأيسر من join ستعود قيمتها فى حالة أنها تطابق الصفوف فى الجدول الأيمن من join.النوع الثانى right outer join عكسه تمام فكل الصفوف للجدول الأيمن من join ستعود قيمتها إذا ما طابقت صفوف الجدول الأيسر من join.النوع الثالث full outer join يعود بكل صفوف الجداول على أيمن وأيسر join.

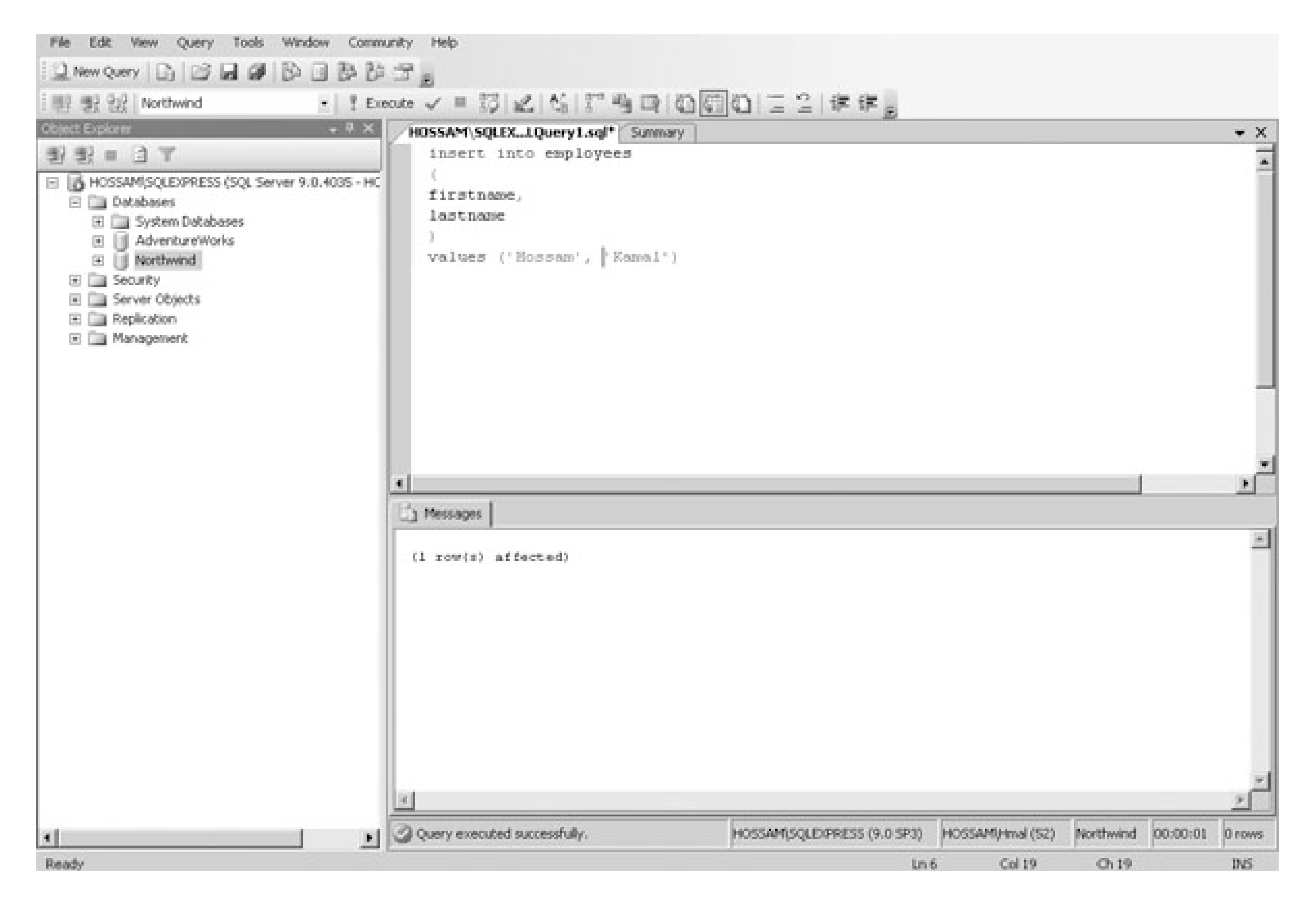
مهلاً ! كل هذا رائع لكن أين الفائدة ؟!
هناك حالات مثل العلاقة Parent – Child مابين الجداول على الرغم من وجود التكامل المرجعي فيما بينهم Referential Integrity بعض صفوف الجداول الأباء Parents لا علاقة لها بصفوف الجداول الأبناء Childs مع احتمالية شغل القيمة null فى أحد صفوفها والتى بمثابة F.K للجدول الأب ولهذا لايحدث matching . لكى نوضح بمثال فإن البيانات الخاصة بكلٍ من الجداول Orders – Employees تحتاج الى بعض الإضافات.
لكى تضيف بيانات موظف employee فإنك ستضيف صفاً جديداً للجدول Employees والذى لايرتبط بعلاقة مع صفوف جدول Orders. لتبسيط الأمر كل ماعليك هو إضافة البيانات الجديدة للأعمدة التى لاتقبل null.

جرب إضافة موظف جديد

افتح نافذة استعلام جديدة ثم أدخل التالي واضغط Execute

```
insert into employees
(
firstname,
lastname
)
values ('Hossam', 'Kamal')
```

ليكون الناتج كما بالصورة التالية



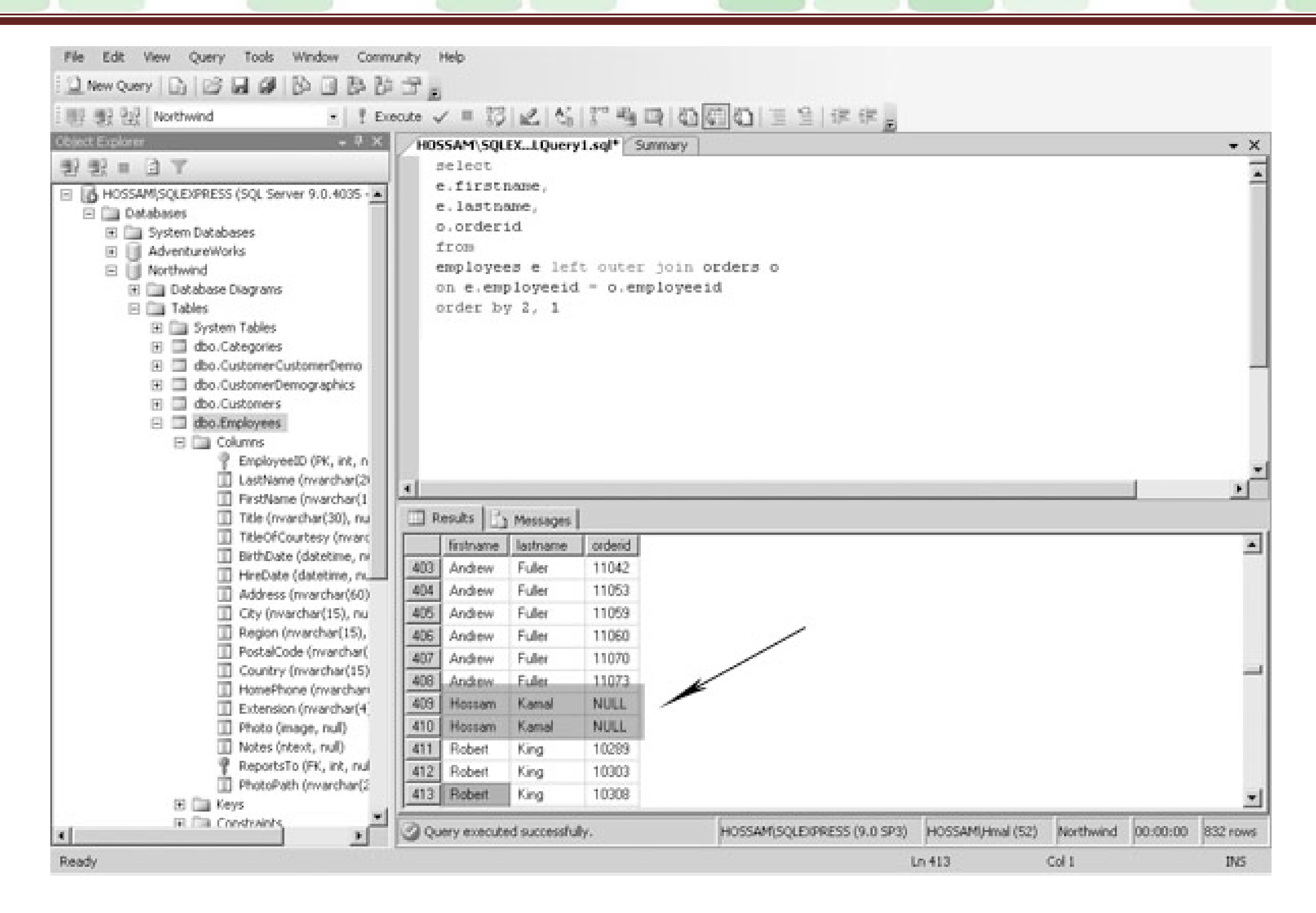
الأمر بسيط فقط جملة INSERT وتعيين العمودين المراد إضافة بيانات لهما مع العلم أن العمود EmployeeID تُضاف قيمته تلقائياً لأنه سبق تعريفه أثناء تصميم الجدول على أنه IDENTITY وباقى الأعمدة تسمح بالقيمة NULL فلا يهم إضافة قيم لها . لديك الآن بعد هذا الاستعلام موظف جديد باسم Hossam Kamal والذى لم يتلق أى order بعد.لنفترض الآن أننا نريد قائمة بالطلبيات orders التى تمت لكل الموظفين على شريطة جلب كافة الـ employees حتى من لم يجرى أى طلبية هذا هو المثال التالي.

جرب LEFT OUTER JOIN بعد تعديل جدول EMPLOYEES

افتح نافذة استعلام جديدة ثم أدخل الاستعلام التالى و F5

```
select
e.firstname,
e.lastname,
o.orderid
from
employees e left outer join orders o
on e.employeeid = o.employeeid
order by 2, 1
```

لترى النتيجة كما بالصورة التالية



إذا ماجربت المثال السابق باستخدام Inner Join فلن تجد الموظف Hossam Kamal ضمن الناتج (جرب هذا بنفسك).
لاحظ فى الاستعلام ان ORDER BY والتى استخدمت لترتيب الناتج على حسب First Name , Last Name والتى استخدمنا فيها ترتيب الأعمدة وليس اسمها (1 ,2) وهى طريقة جيدة لكتابة الاستعلام فى حالة ما إن كان العمود ناتج عن دوال التجميع مثلاً مثل SUM.لاحظ فراغ قيمة OrderID عند الموظف الجديد كما فى الصورة.
لماذا إذاً استخدمنا left outer join وليس right ؟! الإجابة بديهية إذ يُمكنك فعلها فقط بالتبديل مابين أسماء الجداول من الجهة اليسرى إلى اليمنى والعكس وكذلك الـ Correlation names ..وسيكون الناتج واحد أيضاً (جربها بنفسك).

Other Joins

أردت عدم التلميح إلى full outer join لوجود انواع أخرى مثل Union Join و Cross Join وهذه الأنواع قلما تُستخدم وخارج نطاق الكتاب (صراحة لا أحب شرح الأشياء النادر حدوثها إنما أفضل اكتاسبها بالخبرة العملية وقت الحاجة) لكن عموماً سأُعطى لمحات سريعة.
فمثلاً:

Full Outer Join: يعيد لك كافة الصفوف فى الجدولين معاً اليمين واليسار حتى ولو لم يُوجد صفوف مرتبطة معاً.
Union Join: يقوم بإنشاء جدول مستقل عن الجدولين وجمع البيانات فى كلا الجدولين شريطة اتحادهما فى عدد الأعمدة وكذا امكانية قبول نوع بيانات عمود محل الآخر مثلاً اذا كان معرفاً String أو Memo يُمكن احتواءهم.ويكون الاستعلام كالتالى:

```
Select * from table1 union all
Select * from table2
```

Cross Join: وهو يدمج كافة الصفوف من الجدولين وينتج عنه جدول يحوى كافة الأعمدة وصفوف بحاصل الضرب الديكارتي (مصطلح رياضي مشهور) للتوضيح نفترض أن عندنا جدول به خمسة اعمدة وخمسة صفوف وجدول آخر به ثلاثة أعمدة وعشرة صفوف إذا ماقمنا بعمل Cross Join بينهما فإن الناتج جدول ثالث به خمسة عشر صف وثمانية أعمدة (صدقنى لها استعمال لكن خارج تغطيتنا ☺).

خاتمة الفصل

فى هذا الفصل تعلمنا كيفية كتابة استعلامات فى SSMSE واستخدمنا مزايا كثيرة فى SQL ، استعملنا Aggregate Functions تعاملنا مع دوال الوقت والتاريخ ،فهمنا كيفية تطبيق GROUP BY مع أى استعلام ،استعملنا الـ Joins وعرفنا الفروق بينها ، كل هذا وأكثر تعلمناه وكان الغرض أيضًا تنشيطًا لكى تكتب بنفسك حتى وإن استصعبت بعض الاستعلامات فإنك حتما استمتعت بكتابتها ورؤية الناتج بنفسك.

الفصل الثالث

التعامل مع بيانات قاعدة البيانات Manipulating Database Data

سنتعلم فى هذا الفصل التالي:

حان الآن وقت التعديل فى البيانات الداخلية بعد إدخالها عن طريق العمليات Insert , Update ,Delete

- استرجاع البيانات
- استخدام الجملة SELECT INTO
- إداخال البيانات Inserting data
- تحديث البيانات Updating data
- حذف البيانات Deleting data

إسترجاع البيانات

تعلمنا استرجاع البيانات بصورته البسيطة عن طريق استخدام جملة SELECT فى الفصل السابق.فكما نعرف ان البيانات تُخزن فى صفوف rows داخل جداول tables و هذه الصفوف بدورها تحوى أعمدة columns ولكى نبسط الأمر فإن الإستعلام يتكون من جزئين رئيسيين:

- جملة SELECT تُخصص أى من الأعمدة سوف يرجع بالبيانات.
- جملة FROM تُحدد اى جدول / جداول يحوى هذه البيانات.

*تنويه: هل لاحظت استخدامى جملة SELECT و FROM بالحروف الكبيرة Capital.. هذا لا يعنى أنك لابد ان تفعل ذلك أثناء كتابتك لأى استعلام. فلغة SQL **ليست** حساسة لحالة الأحرف Case sensitive وغالباً الكلمات المحفوظة keywords تُكتب بالحروف الصغيرة lowercase . فى لغة T-SQL التى تُستعمل لإجراء الاستعلامات فى MS SQL SERVER يُسمى الاستعلام SELECT لكن فى ISO/ANSI وهى المنظمة المخول لها وضع معايير قياسية للغات البرمجة تُفرق بين الاستعلام والجمل queries and statements فالاستعلام ينتج عنه جدول ويتم على جدول أما الجمل statements قد/وقد لا تعمل على جدول ولاينتج عنها جدول كنتيجة، هذا كان تنويه لعلك تُسمى الـ query باسم SELECT سمها ماشئت مادمت تفهم قصدى خلال الكتاب.*

استخدم الكلمتين SELECT , FROM لتجلب البيانات من أى جدول كما بصورتهم البسيطة فى التالي:

```
Select * from <table name>
```

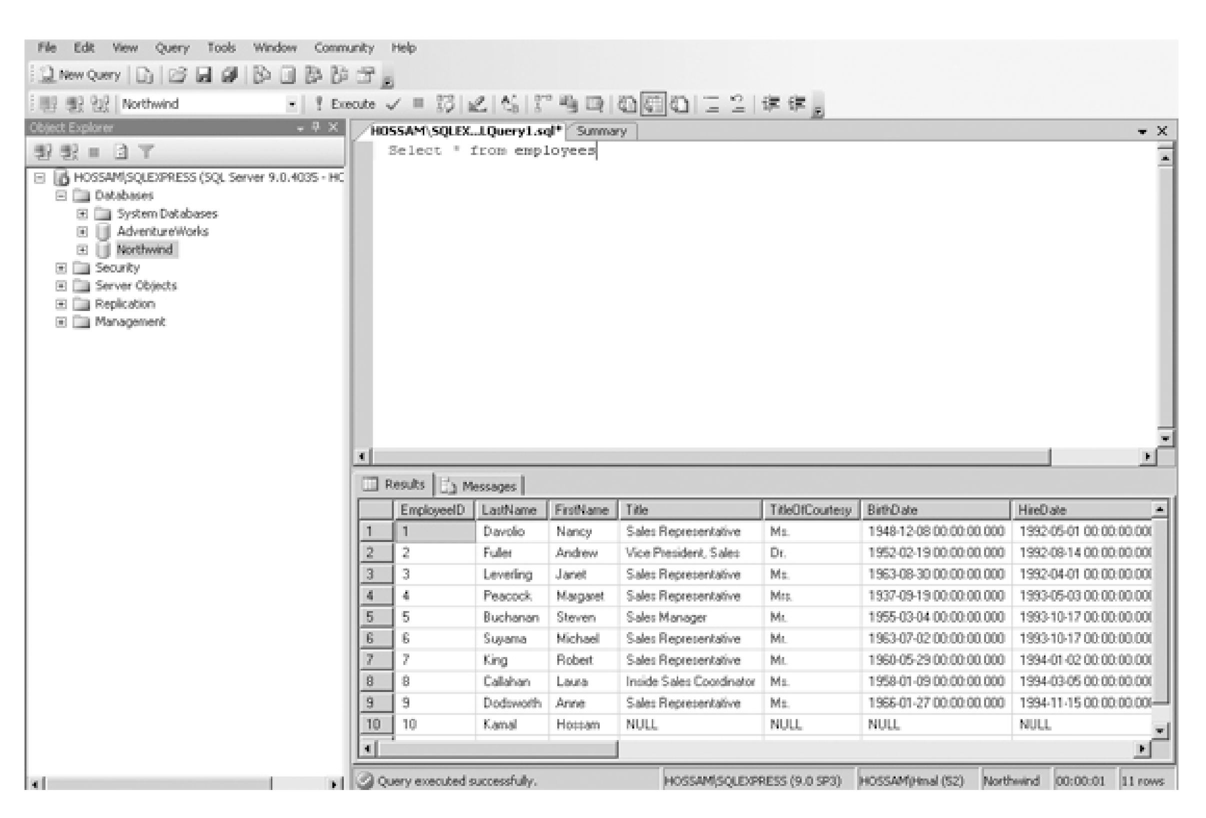
كما تعلم فإن (*) asterisk تعنى كل الأعمدة فى الجدول .

إجراء استعلام بسيط

افتح SSMSE تأكد أن قاعدة البيانات التى تعمل عليها هى Northwind أدخل الإستعلام التالي ثم Execute

```
Select * from employees
```

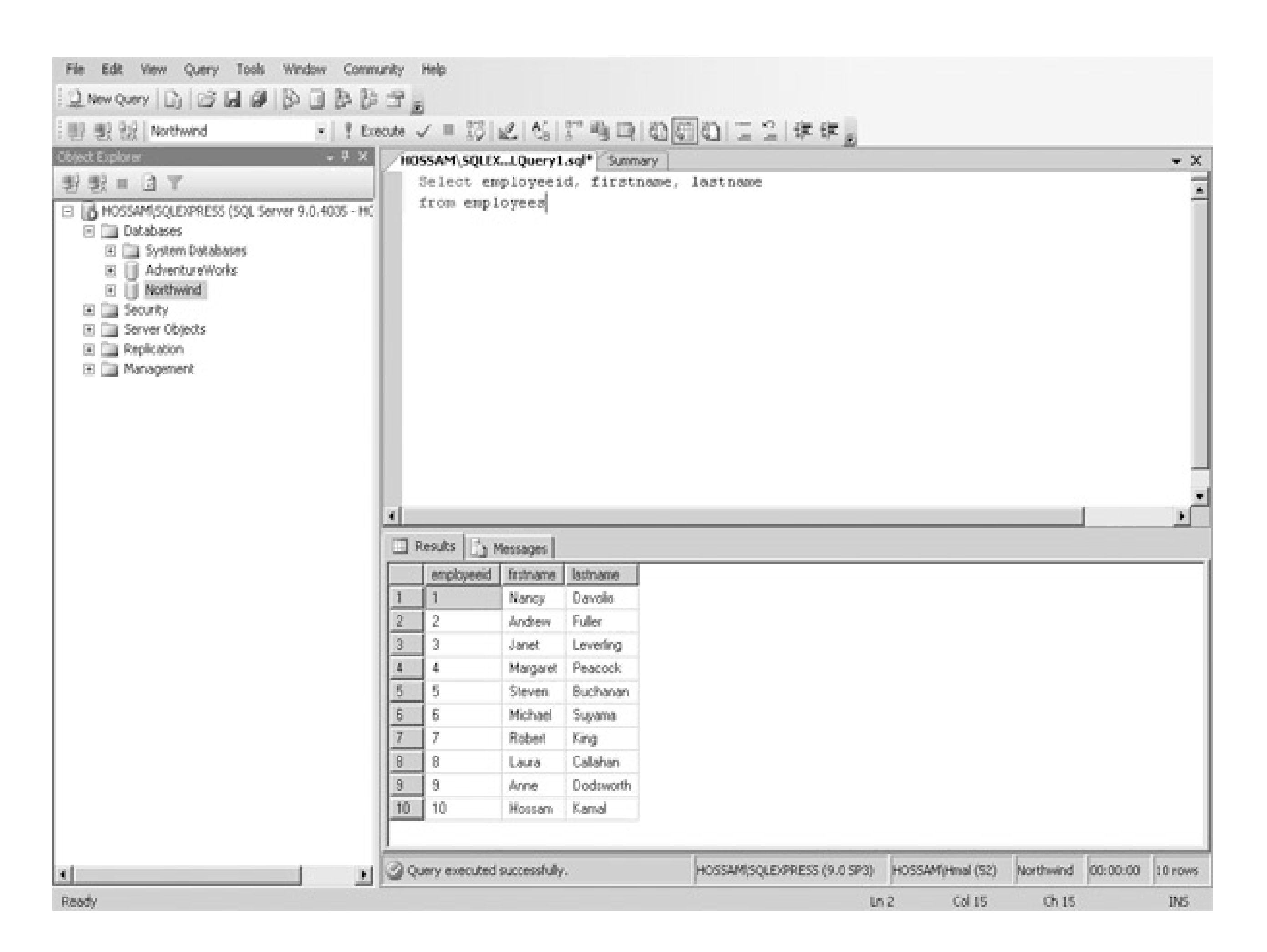
لترى النتيجة كما بالشكل



ماحدث انت تعرفه ، ببساطة أنت طلبت الحصول على كافة أعمدة جدول الموظفين Employees وكان ذلك ! ولتتأكد من أنك حصلت على كافة الأعمدة يُمكنك أن تتصفح الجدول الناتج بالإزاحة ناحية اليمين لتراهم جميعاً.
هناك ملحوظة يُرجى التنبه لها ..فحينما تجرى استعلاماً ما تأكد من حصر المطلوب على أعمدتك فقط وذلك حتى لاتُضيع الموارد Resources بلا فائدة ، ولعمل ذلك اتبع الآتى :
افتح نافذة استعلام جديدة New Query وأدخل الاستعلام التالي ثم F5..

```
Select employeeid, firstname, lastname
from employees
```

هذا الاستعلام يُرجع لك ثلاثة أعمدة فقط هى EmployeeID,FirstName,LastName من الجدول Employees.
وتشاهد النتيجة كما بالشكل ..



استخدام الشرط WHERE

كما رأينا فى الفصل السابق أن الشرط WHERE يقوم بتحديد مرادك تماماً من الاستعلام عن طريق شروط مابين منطقية أو مقارنة بين قوالب Patterns كما رأينا مُسبقاً.
فى أبسط صوره يتكون الشرط WHERE من التالي..

WHERE <column1> <operator> <column2 / Value>

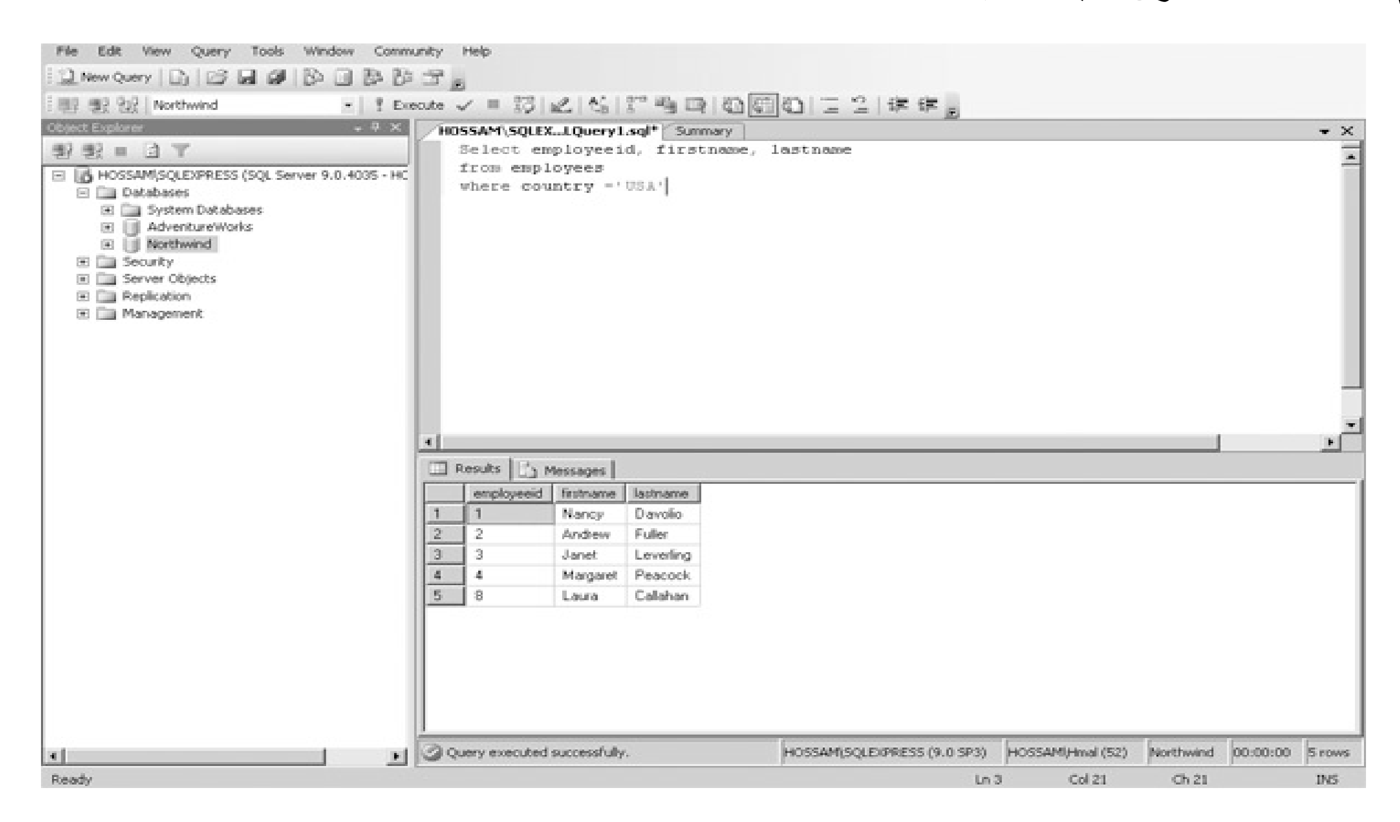
<operator> يُحتمل إحدى عوامل المقارنة التالية (= و < > و < و > على سبيل المثال) ووجه المقارنة إما عمود آخر أو قيمة value.

لتحسين الاستعلام السابق سنستخدم WHERE

أضف السطر التالى للإستعلام السابق (أرجوا أن تكون لازلت فاتحاً SSMSE):

Where country = 'USA'

ثم اضغط Execute لترى النتيجة كما بالشكل ..



تنوية آخر: كما أسلفت فإن SQL ليست حساسة لحالة الأحرف ، ولكن سلسلة الحروف string literals والتي توضع بين ' quotes' حساسة لحالة الحرف ماكان Capital أو small ولهذا قمنا بكتابة USA بالحروف الكبيرة فيما غير ذلك قد لا تحصل على نتيجة فانتبه .

ماحدث واضح فجملة *where* حددت لنا المطلوب بدقة و هم الموظفين أصحاب البلد *'USA'* وتم لنا المراد برجوع جدول به أسماء الموظفين وكذا الـ *ID* الخاص بهم.

استخدام عوامل المقارنة Comparison Operators مع الشرط WHERE

لاحظ الجدول التالي :

المُعامل	الوصف	مثال
=	يُساوي	EmployeeID = 1
<	أقل من	EmployeeID < 1
>	اكبر من	EmployeeID > 1
<=	أقل من أو يُساوي	EmployeeID <= 1
>=	اكبر من او يُساوي	EmployeeID >= 1
< >	لايُساوي	EmployeeID < > 1
!=	لايُساوي	EmployeeID != 1
!<	ليس أقل من	EmployeeID !< 1
!>	ليس أكبر من	EmployeeID !> 1

إذا عملت من قبل على *SQL* اكن مع منصات أخرى غير مايكروسوفت فانتبه لأن ليس كل هذه المعاملات فى الجدول موجودة فى أى اصدار آخر إلا فى *T-SQL* والتى نلتزم بالـ *Syntax* الخاص بها طوال الكتاب.
بالإضافة لهذه المُعاملات يُوجد أيضاً المُعامل *LIKE* والذي تطرقنا له سابقاً على عجالة (سنناقشه لاحقاً فى الفصل التالي) لكن لا مانع من جدول يحتوي بعض التفاصيل:

المُعامل	*الوصف*	*مثال*
LIKE	تحيد Pattern معينة للمقارنة	Where Title LIKE 'Sales%' وتعنى اختيار العمود Title الذى يبدأ بكلمة Sales متبوعة بأى عدد من الحروف

هذا المعامل بدوره يحتوى بعض الـ Wildcards والتى نُوضحها فى الجدول التالى:

الرمز	*الوصف*
%	ويحل محل أى عدد من الحروف سواءاً اتى قبل أو بعد الكلمة مثال: where firstname like 'Ho%' حينها يعود بنتيجة مثل Hossam,Hosney ...إلخ
_	يحل محل حرف واحد فقط مثال : where title like '_ales' لتعود بنتيجة مثل Aales,bales......إلخ
[ ]	يحل محل حرف واحد فقط فى range مُعين بين القوسين مثل [a-f] أو [abcd]. مثال: where title like '[b-g]ales' فمن الممكن أن تكون النتيجة bales , fales ولا تتعدى بين القوسين.
[^]	وتعنى حرف واحد ليس من ضمن المحدد بين القوسين مثل [^a-g] او [^abcde]

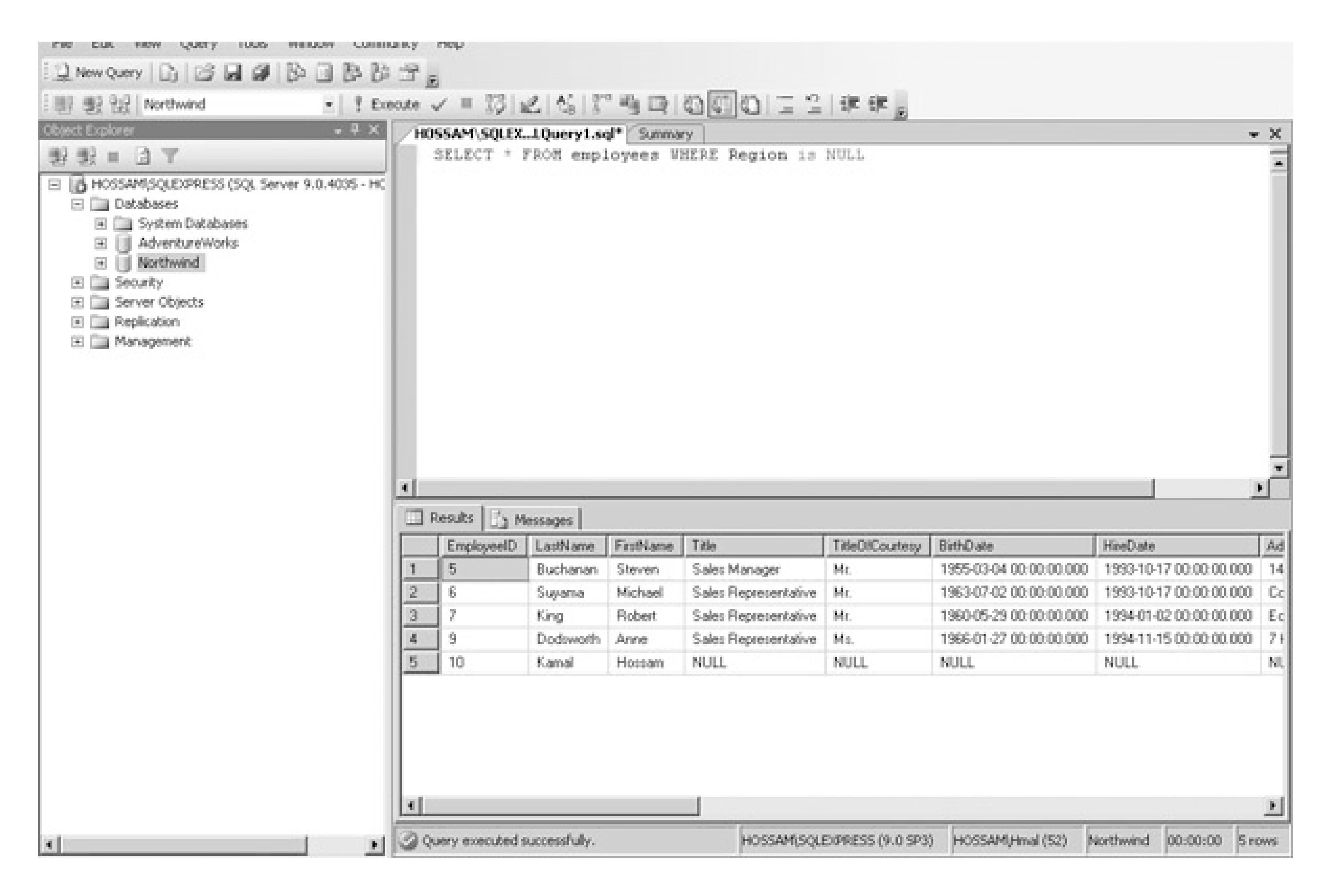
فى بعض الأحيان تكون القيمة فى بعض الصفوف غير معلومة وأنت تحتاجها فى استعلامك.وهناك أيضاً أعمدة تحوى القيمة null نتيجة عدم اسناد أى بيانات لها (ليس معنى ان قيمة ما فى عمود =صفر أنه NULL)، ولعمل إستعلام على هذه الأعمدة نحتاج الشرط IS [NOT] NULL انظر الجدول التالي:

المُعامل	*الوصف*	*مثال*
IS NULL	تسمح لك بجلب بيانات الأعمدة التى لا تحوى قيمة	Where region IS NULL
IS NOT NULL	تسمح لك بجلب بيانات الأعمدة والتى تحتوى على قيمة حتى ولو كانت (٠)	Where region IS NOT NULL

ستقول لى من الممكن الاستغناء عن هذين الشرطين باستخدام أحد المعاملات السابقة مثل = أو=!..
وهذا خطأ شائع فمثلاً اكتب هذا الاستعلام

```
SELECT * FROM employees WHERE Region = NULL
```

اضغط F5 لترى النتيجة وقد خرجت لك بجدول فارغ يحوى أسماء الأعمدة فقط ...لكن اذا استبدلنا = بالشرط IS ستر النتيجة وقد عادت لك بصفوف للعمود region ولكن الذى لايحوى قيمة انظر الشكل التالي..



ولإجراء استعلام اختيار قيمة تقع فى range معين نستخدم الشرط BETWEEN و IN ولتوضيع عملهما انظر الجدول التالي..

المُعامل	الوصف	مثال
BETWEEN	إذا كانت القيمة مابين الـ range المُحدد تعود بـ True	Where extension BETWEEN 400 AND 500 تعود بالصفوف بين القيمتين المحددتين
IN	إذا كانت القيمة موجودة ضمن الشرط تعود بـ True	Where city IN ('Seattle','London') تعود بالصفوف التى تحوى إحدى القيمتين

دمج الشروط Combining Predicates

فى بعض الأحيان قد تحتاج إلى شروط متداخلة لتحقيق استعلامك تماماً ولفلترة بياناتك ، يُمكنك هذا باستخدام المُعاملات المنطقية Logical Operators ولتوضيح ذلك انظر الجدول التالي:

المُعامل	الوصف	مثال
AND	يقوم بدمج عبارتين ويعود بالقيمة True فقط اذا كانت كلا العبارتين True	Where (title LIKE 'Sales%' AND lastname = 'Peacock')
NOT	يقوم بنفي قيمة منطقية	Where NOT (title LIKE 'Sales%' AND lastname = 'Peacock')
OR	تدمج عبارتين وتعود بالقيمة True إذا كانت إحداهما صحيحة	Where (title = 'peacock' OR title ='King')

من المُستحسن استخدامك الأقواس حتى يسهُل قراءة استعلامك ..وفى بعض الاستعلامات المتقدمة يكون حتم عليك هذا.

فرز البيانات Sorting Data

بعد قيامك بفلترة البيانات التى تريدها، بإمكانك فرز البيانات قياساً على احد الأعمدة او اكثر من عمود صعوداً أو هبوطاً ، فالنتائج العادية التى تحصل عليها من الجدول تكون فى الغالب غير مرتبة وغير معلوم طريقة ترتيبها. ولكى تجبر النتائج على ترتيب معين فإننا نستخدم الجملة ORDER BY بهذا الشكل :

```
ORDER BY <column> [ASC | DESC] {, n}
```

يُعبر الشرط <column> عن العمود الذي سيتم الترتيب حسب ما به من قيمة.والعبارة {, n} تعنى تخصيص أى رقم من الأعمدة سيُطبق عليه الشرط مفصولاً بالفاصلة مثل {2,5,7,15} وبالطبع سيتم الترتيب صعوداً أو هبوطاً طبقاً للشرطين ASC|DESC.

- ASC: Ascending (1, 2, 3, 4, and so on
- DESC: Descending (10, 9, 8, 7, and so on

ولك أن تعلم أنه فى حالة نسيانك وضع شرط لترتيب الاستعلام الناتج فإن القيمة الافتراضية ستكون تصاعدياً ASC .

المثال التالى هو الشكل العام لهذا النوع من الاستعلامات:

```
SELECT <column>
FROM <table>
WHERE <predicate>
ORDER BY <column> ASC | DESC
```

ستقوم الآن بتطبيق القاعدة العامة بأعلى على مثال حقيقي ، ففى هذا المثال سنقوم بالتالي:

- استرجاع قيمة الطلبيات orders التى تمت بواسطة الموظف رقم ٥ (employee 5) .
- استرجاع الطلبيات المباعة الى France أو Brazil.
- إظهار الأعمدة OrederID , EmployeeID ,CustomerID, OrderDate, ShipCountry .
- سيتم الفرز طبقاً للدولة المباع لها وتاريخ البيع.

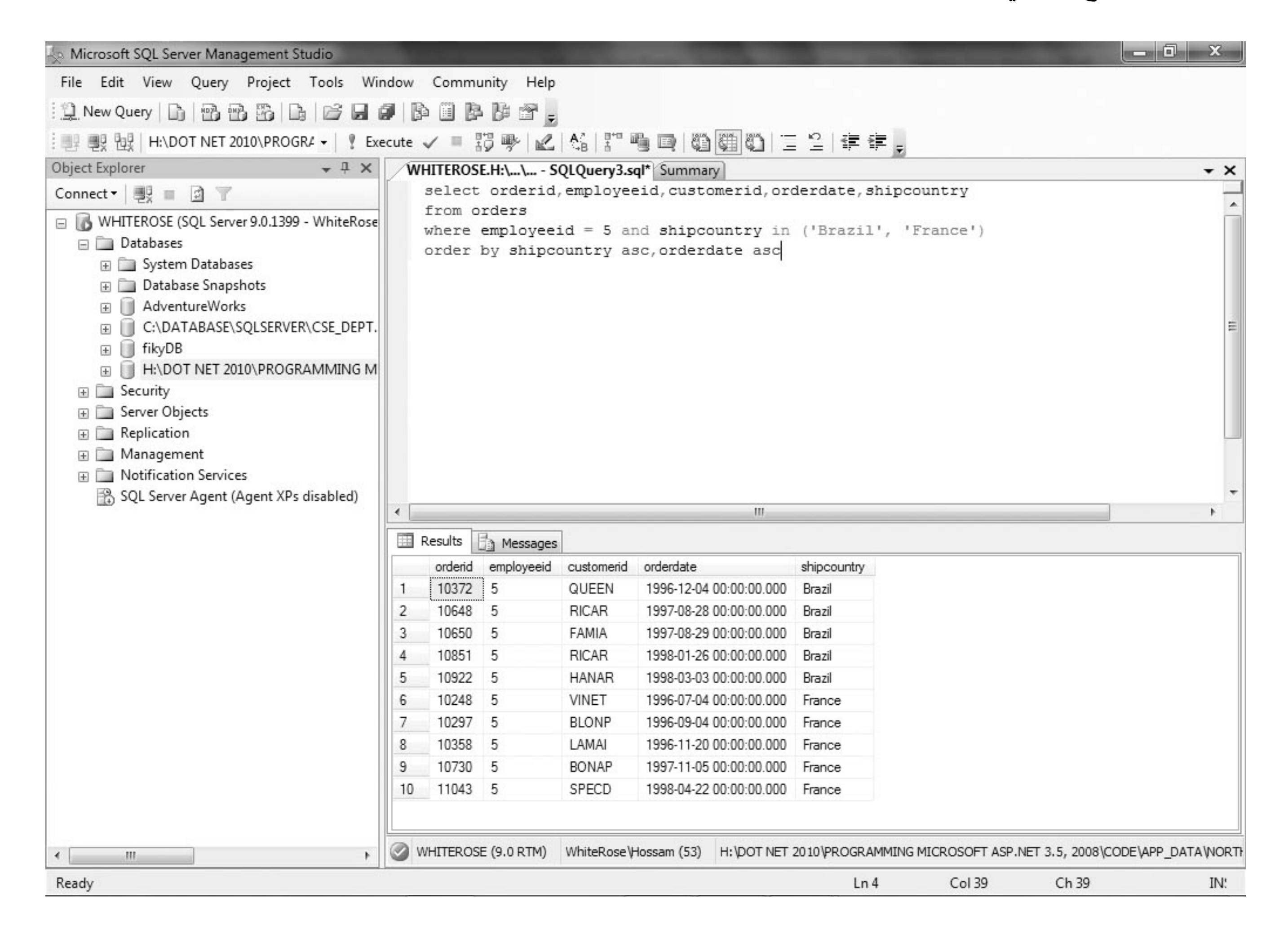
هل يبدو الأمر معقداً ؟ ☺ لا تيأس فقط حاول.

افتح نافذة استعلام جديدة فى SQL Server Management Studio .

أدخل الاستعلام التالى ثم اضغط F5 :

```
select orderid,employeeid,customerid,orderdate,shipcountry
from orders
where employeeid = 5 and shipcountry in ('Brazil', 'France')
order by shipcountry asc,orderdate asc
```

ليظهر لك الناتج كالتالي:



كيف حدث هذا

انظر إلى كل جملة منفصلة.فمثلا الجملة SELECT تُخصص الأعمدة المُستخدَمة

select orderid,employeeid,customerid,orderdate,shipcountry

واما الجملة FROM فيُعنى بها الجدول الذي سنقوم بعمل الإجراءات عليه

from orders

وأما الشرط المتمثل فى الجملة WHERE فهو مايحتاج لقليل من التفكير حتى تصل للإجراء المنشود:

- رقم الموظف EmployeeID يساوى ٥
- ShipCountry لابد أن تكون Brazil أو France

where employeeid = 5 <u>and</u> shipcountry in ('Brazil', 'France')

لاحظ وجود الرابط المنطقي AND وهو مايعني أنه لابد من تحقيق الشرطين معاً.

أما الشرط ORDER BY فهو المسئول عن ترتيب الناتج من الصفوف وهو كما فى مثالنا سيتم ترتيبه أولاً على جهة الشحن ShipCountry ثم على تاريخ الطلب OrderDate

order by shipcountry asc,orderdate asc

إستخدام الجملة SELECT INTO

تُستخدم هذه الجملة لإنشاء جدول جديد يحتوي أو لا يحتوي كم الناتج العائد من جملة SELECT، فهى تنسخ الـ Structure الخاص بالجدول وبياناته إلي جدول آخر مُخصص بعد الجملة INTO.باختصار تعمل على أخذ نسخة احتياطية من أى جدول BackUp .

إذا تم تضمين العلامة (#) hash إلى اسم الجدول ، فهذا يعنى إنشاء جدول مؤقت فى قاعدة البيانات System (tempdb) بغض النظر عن أى قاعدة تعمل عليها الآن . ماذا لو لم ندرجها ؟! ...سيتم انشاء جدول مماثل ولكن فى قاعدة البيانات التى تعمل عليها وليست tempdb.

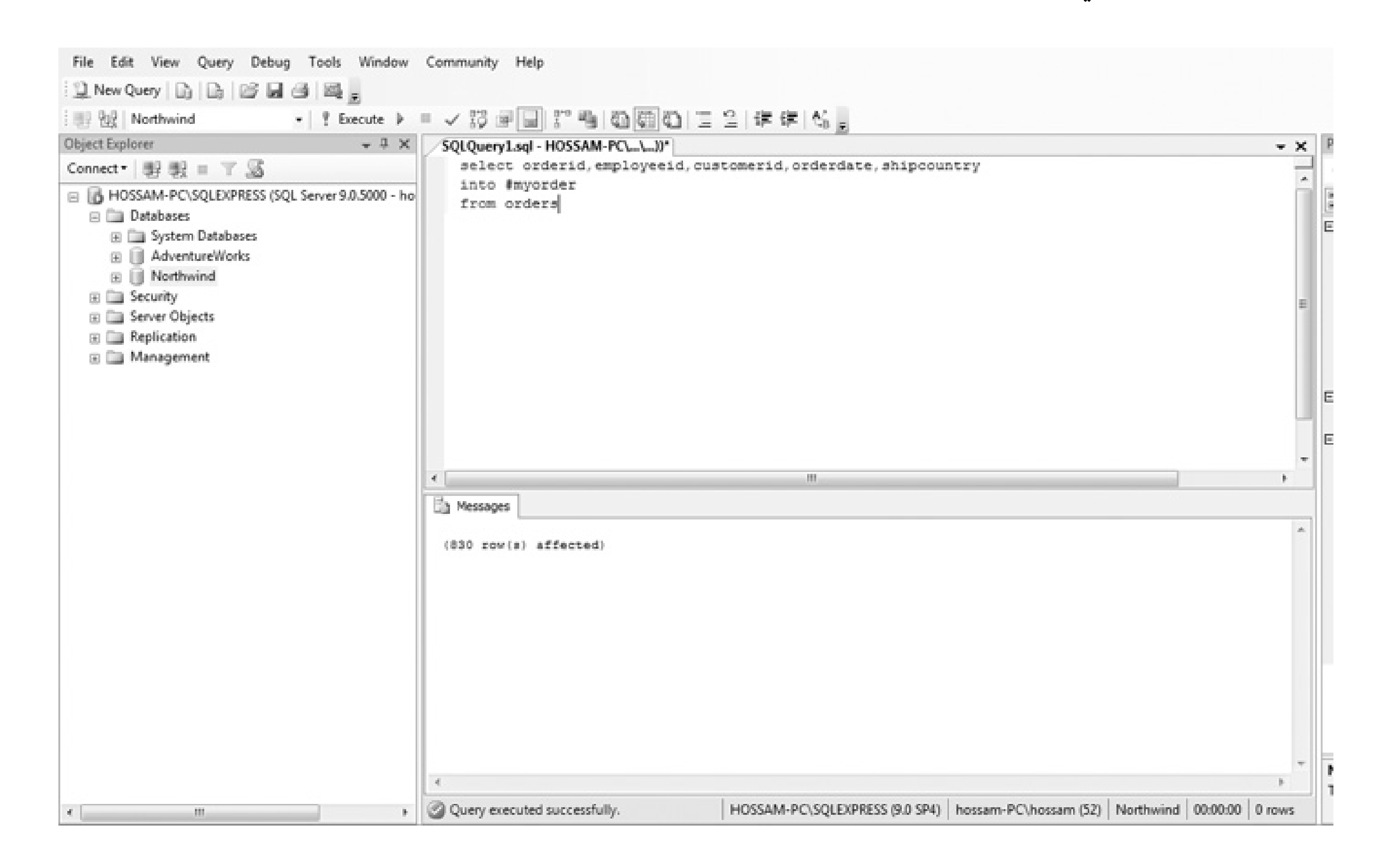
لك ان تعرف أيضاً أنه وبمجرد انشاء الجدول الجديد فإنه سيأخذ كل صفات الجدول الأصلي والشروط الموضوعة على الأعمدة مثل احتواء القيمة null وكذلك data types ومع ذلك فإنه لايتم نسخ أي constraints أو مفهرسات indexes ولا حتى triggers.

لتجريب ذلك اتبع الآتي:-

افتح نافذة استعلام جديدة وتذكر انك تستخدم قاعدة البيانات NorthWind أدخل الإستعلام التالي ثم اضغط F5 :

```
select orderid,employeeid,customerid,orderdate,shipcountry
into #myorder
from orders
```

لترى كما فى الشكل التالي:

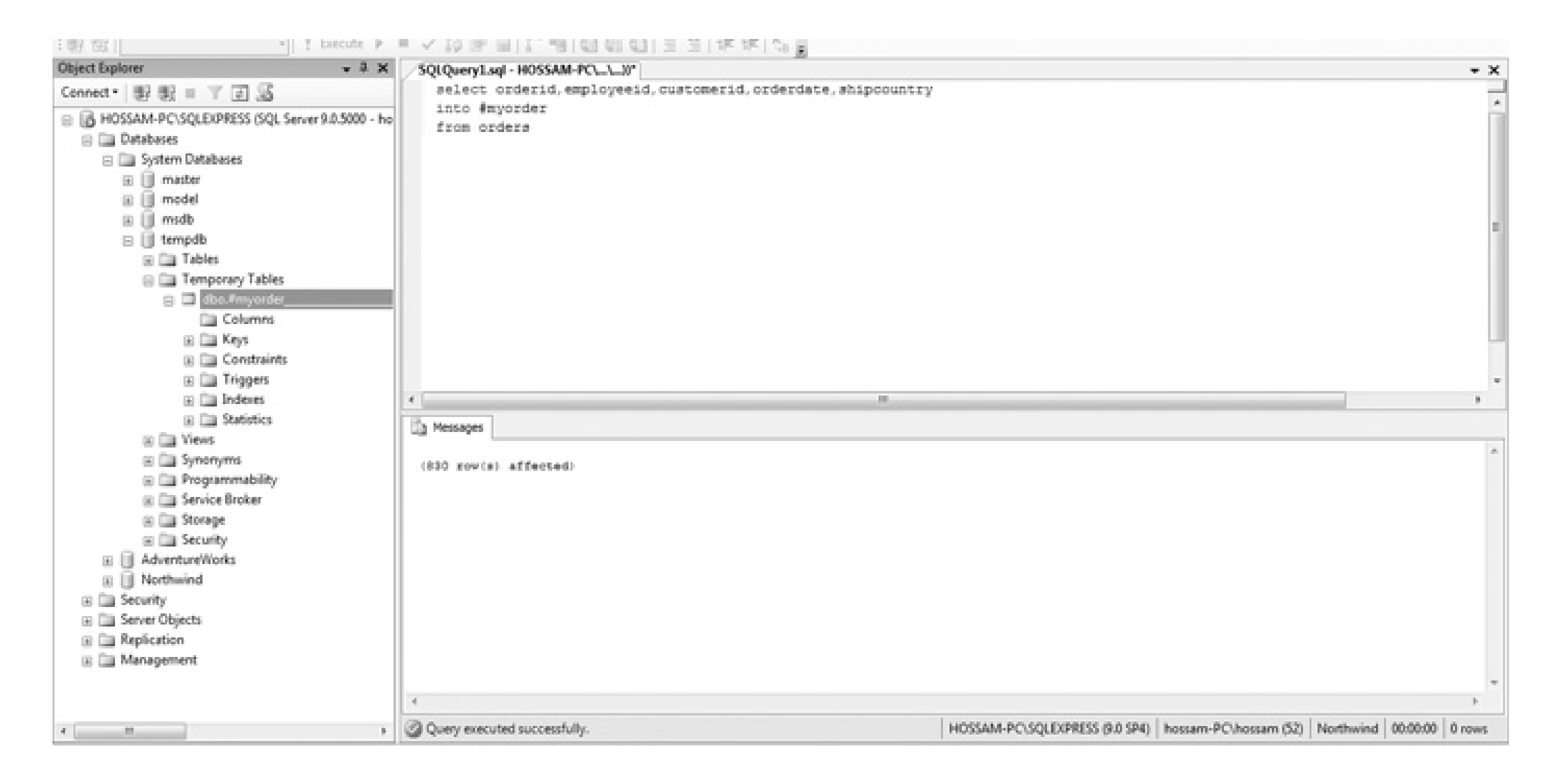


لشرح ماحدث :

قمنا باستخدام الجملة SELECT وذلم لتعيين قائمة الأعمدة المراد وضعها فى الجدول الجديد ، ثم تعيين اسم للجدول الجديد ووضعه فى الـ tempdb بالجملة INTO #myorder ثم أتبعناها باسم الجدول الأصلي from orders ، وحتى لو كنا نعمل على قاعدة البيانات Northwind تذكر ماقلناه بأن الجدول المسبوق بـ (#) يتم تخزينه فى قاعدة البيانات المؤقتة tempdb ، وهذه الجداول تبقى مُخزنة فى قاعدة البيانات المؤقتة مادمت لاتزال فاتحاً نافذة الاستعلام ، وبمجرد إغلاقك إياها حتى ولو قمت بعمل حفظ فإنه سيتم حذفها تلقائياً من القاعدة tempdb.

بمجرد انشاءك لهذا الجدول أصبح بالامكان استخدامه كأي جدول معتاد دون أى مشاكل ، تنبه أيضاً أن هذه الجداول المؤقتة تُحذف بمجرد غلقك لـ SSMSE وذلك لأن قاعدة البيانات المؤقتة يُعاد بناؤها فى كل مرة يتم فتح وغلق SSMSE.

لكى تري أين تقبع هذه الجداول انظر إلي الشكل التالي:



ولكي ترى إمكانية اجراء استعلامات عليها كما هو المعتاد مع الجداول الأخرى انظر الى الشكل التالي:

```
select * from #myorder
```

	orderid	employeeid	customerid	orderdate	shipcountry
1	10248	5	VINET	1996-07-04 00:00:00.000	France
2	10249	6	TOMSP	1996-07-05 00:00:00.000	Germany
3	10250	4	HANAR	1996-07-08 00:00:00.000	Brazil
4	10251	3	VICTE	1996-07-08 00:00:00.000	France
5	10252	4	SUPRD	1996-07-09 00:00:00.000	Belgium
6	10253	3	HANAR	1996-07-10 00:00:00.000	Brazil
7	10254	5	CHOPS	1996-07-11 00:00:00.000	Switzerland
8	10255	9	RICSU	1996-07-12 00:00:00.000	Switzerland
9	10256	3	WELLI	1996-07-15 00:00:00.000	Brazil
10	10257	4	HILAA	1996-07-16 00:00:00.000	Venezuela
11	10258	1	ERNSH	1996-07-17 00:00:00.000	Austria
12	10259	4	CENTC	1996-07-18 00:00:00.000	Mexico

Query executed successfully.

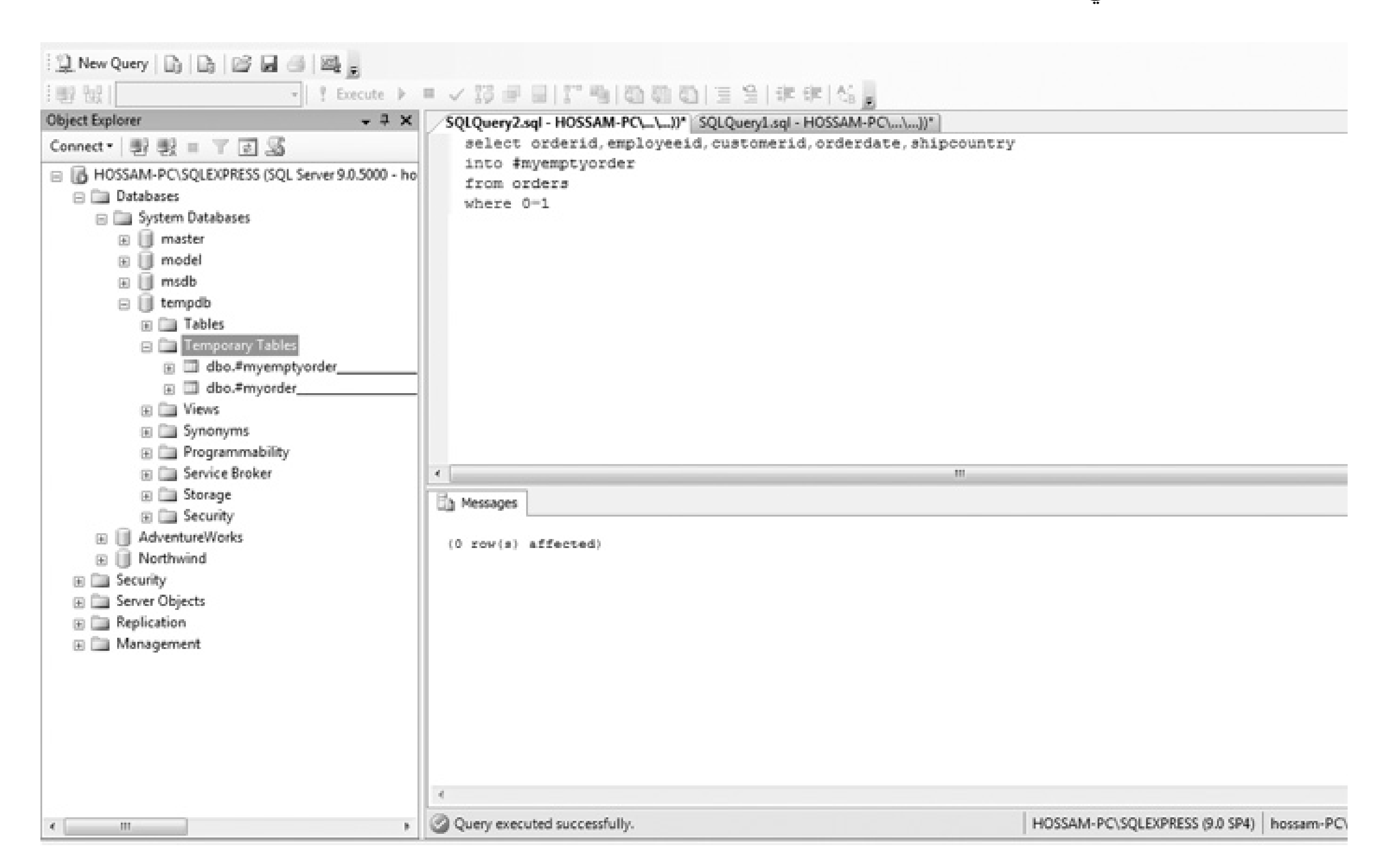
استخدام SELECT INTO لنسخ Table Structure

فى بعض الأحيان نحتاج إلى جدول ما ولكن بدون المحتوي الخاص به ، كان نحتاج أسماء الأعمدة وصفاتها أو كأننا نريد تفريغ الجدول من بياناته ، ولعمل ذلك فلابد من إدارج شرط لايرجع بالقيمة true .وبعدها فانت حر فى ادراج البيانات التى تريدها.

لعمل ذلك ، ادخل الاستعلام التالي فى نافذة استعلام جديدة على قاعد البيانات Northwind:

```
select orderid,employeeid,customerid,orderdate,shipcountry
into #myemptyorder
from orders
where 0=1
```

لتشاهد كما بالشكل التالي :



هل لاحظت الشرط الغريب where 0=1 ؟!

إنه شرط لن يتحقق وهو ما ذكرناه (بإمكانك تخصيص أى شرط لن يتحقق كما تشاء) وبهذه الطريقة فإن table structure يتم نسخها فى قاعدة البيانات المؤقتة tempdb ولكى ترى الجدول الناتج اذهب كما فى الصورة ، ثم قف على المجلد Temporary Tables ثم اضغط على الزر refresh فى الأعلى أو بالزر الأيمن واختر refresh .

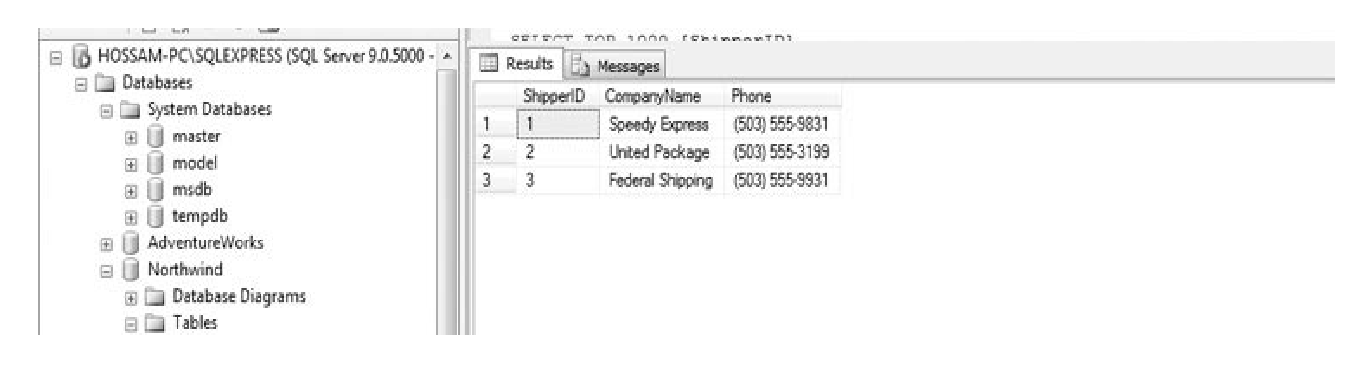
وكما سترى فإن الجدول يحوى الهيكل فقط دون بيانات وللتأكد من ذلك فقط قم بعمل جملة SELECT على الجدول لترى كما بالشكل أسماء الأعمدة وفقط من دون أيه بيانات :

إدخال البيانات Inserting Data

إن من أهم الإجراءات التى تتم على البيانات هى عملية إدخال البيانات إلى الجداول، ونقوم بهذا باستخدام الجملة INSERT . إن هذه الجملة غاية فى البساطة عن أى جملة أخرى ، ذلك لعدم فاعلية استخدام أية عبارات شرطية مثل WHERE – ORDER BY ، وتتكون جملة INSERT من الآتي:

```
INSERT INTO <table>
(<column1>, <column2>, ..., <columnN>)
VALUES (<value1>, <value2>, ..., <valueN>)
```

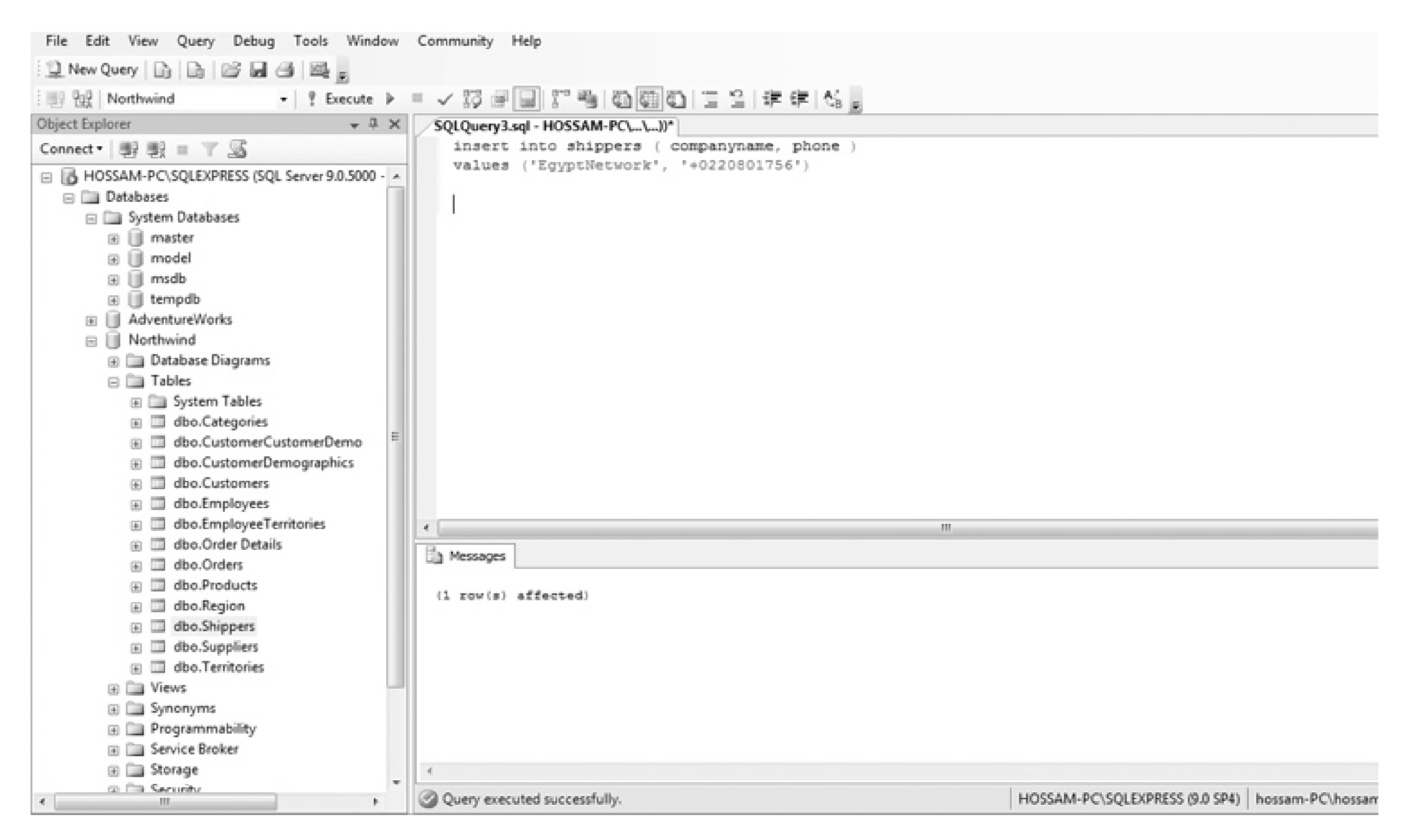
ولتفعيل هذه الجملة نضعها فى المثال التالي ، وهو إضافة صف جديد الى الجدول Shipper فى قاعدة البيانات Northwind . لننظر إلى الجدول أولاً، افتح SSMSE ثم تصفح التسلسل الشجري على اليسار إلى ان تصل الى قاعدة البيانات Northwind ثم اضغط العلامة (+) حتى تصل الى الجدول Shippers ثم اضغط كلك يمين ثم اختر فتح الجدول Open Table كما ترى فى الشكل فإن الجدول لايحوى سوى ثلاثة صفوف :



والآن نأتى لإدراج صف جديد بالجملة Insert
افتح نافذة استعلام جديدة ثم أدخل الإستعلام التالي واضغط F5:

```
insert into shippers ( companyname, phone )
values ('EgyptNetwork', '+0220801756')
```

لترى النتيجة كما فى الشكل التالي:



وتظهر رسالة كما بأسفل تخبرك (1 row (s) affected) .
لشرح ذلك ، لربما تساءلت اولاً لماذا أدخلنا قيمتين بدلاً من ثلاث (هذا اذا كنت قوي الملاحظة ☺)؟
والجواب هو أن القيمة التى لم ندخلها هنا هى ShipperID وهى بمثابة Identity للعمود وتم تخصيص نوعها لتُولد تلقائيا دون تدخل من المستخدم حيث يقوم SQL Server بهذه الوظيفة نيابة عنك ، لذا فإننا فقط سنقوم بادخال قيمتين هما (Companyname ,phone) وذلك حتى لايحدث مشاكل غير متوقعة لم نترك الاختيار لـSQL Server وقمنا بتعينهما فى الاجراء ثم قمنا بتعيين القيم المخصصة لهما بالترتيب values ('EgyptNetwork', '+0220801756').

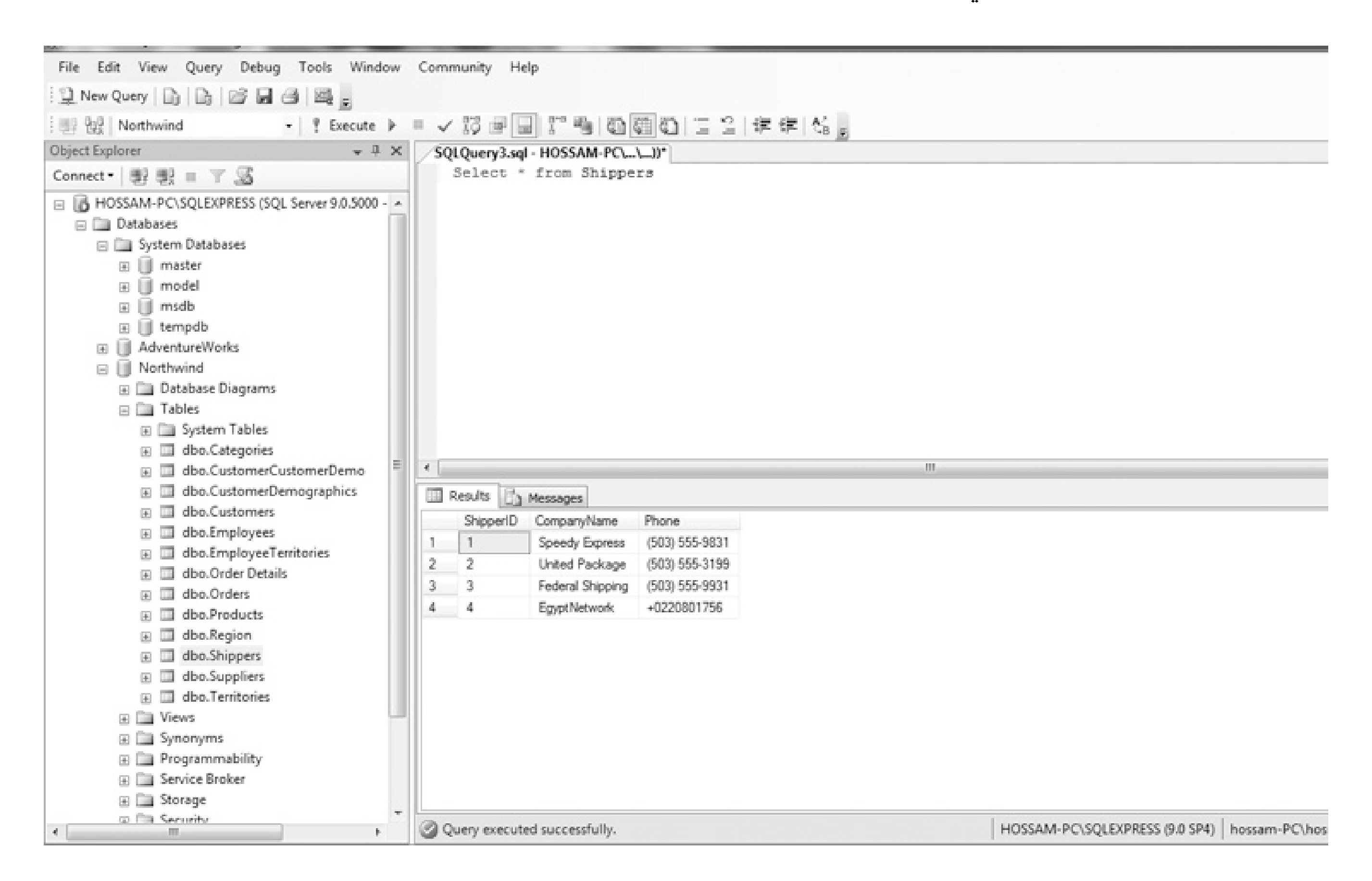
ملاحظة:

هناك بعض القيود على الجملة INSERT فمثلا عندما نريد ادخال قيمة على جدول ثانوي لجدول أساسي foreign key table وهذا الجدول الأساسي ليس فيه Related Parent Record فحتماً سيظهر أخطاء وذلك لأنه لابد من وجود هذه القيمة فى الجدول الأساسي أولاً، على سبيل المثال الجدول Shoppers سنعتبره P.K table للجدول Orders الذي يحوى عمود F.K يُسمى ShipVia والذى بدوره يقوم بعنونة العمود ShipperID فى الجدول Shippers (فقط أعد القراءة لتستوعب ☺) فلن تستطيع اذاً أن تدخل قيمة إلى الجدول Orders قبل ان تدخلها فى الجدول Shippers.

نعود إلى مثالنا ، للتأكد من أنه تمت اضافة القيمة السابقة للجدول ،بديهي أن تقوم بعمل استعلام على الجدول لنرى القيمة ولا أرى أفضل من الاستعلام * SELECT للجدول كما بالتالي:

```
select * from Shippers
```

ليظهر عندك النتيجة كما بالشكل التالي :



كما ترى فقد تم إضافة صف جديد إلى الجدول ، تنبه أثناء إدخالك للبيانات أن تكون من نفس Data Type للعمود المزمع إدخال قيمة له ، فكلا العمودين السابقين من النوع Character data type ،فلو كان أحدهما من النوع Integer مثلا فلن يقبل القيمة النصية ولابد من ادخال قيمة رقمية.

تحديث البيانات Updating Data

ماذا عن تحديثك لبيانات تم ادخالها بالفعل ؟ هذا هو ما سنقوم به باستخدام الجملة UPDATE. عندما نستخدم هذه الجملة لابد من توخى الحذر مع جملة الشرط WHERE وإلا فإن التحديث سيكون لجميع الصفوف فى الجدول.
للتعرف على هذه الجملة إليك قاعدة كتابتها :

```
UPDATE <table>
SET <column1> = <value1>, <column2> = <value2>, ..., <columnN> = <valueN>
WHERE <predicate>
```

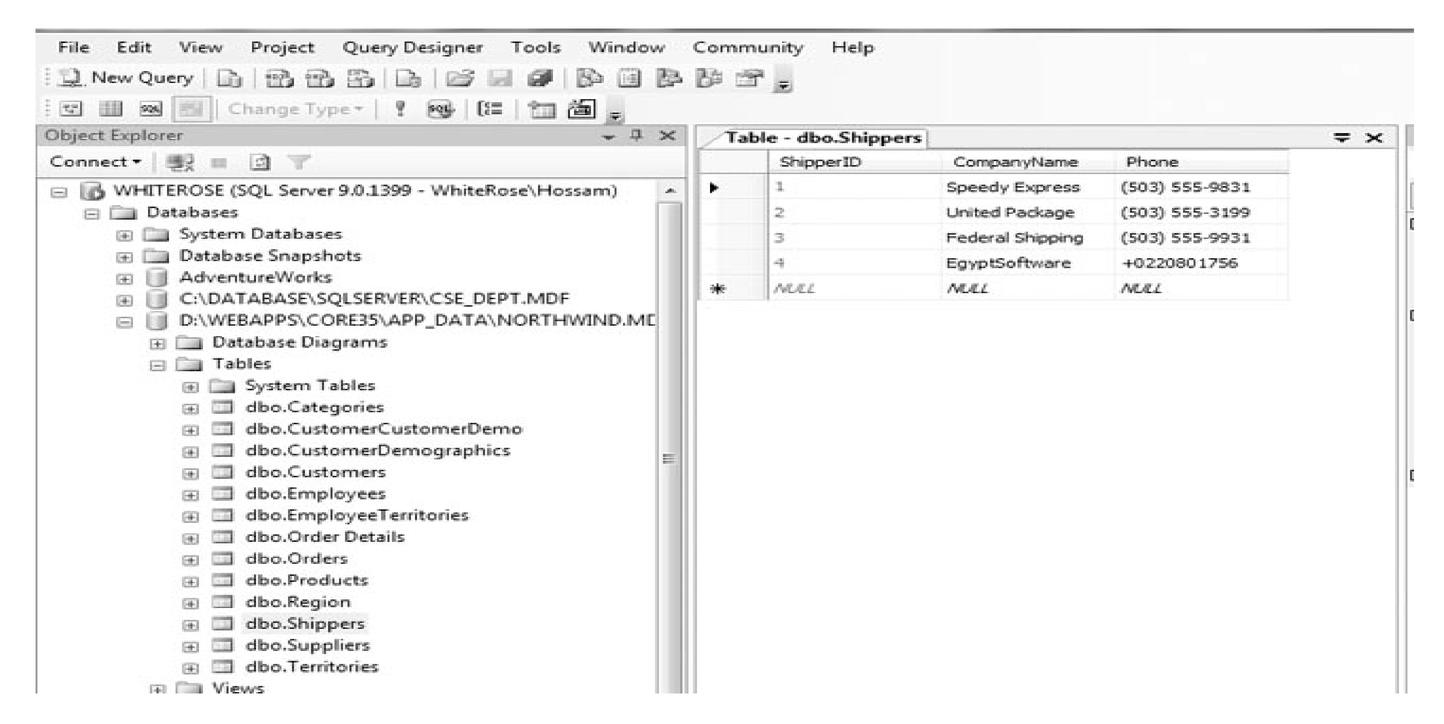
كمثال عملي، لنفترض أن الشركة EgyptNetwork قررت تغيير اسمها إلى EgyptSoftware مثلاً،لكي نجرى هذا التغيير فى قاعدة البيانات ينبغى تحديد اى من الصفوف سيتم اجراء التعديل عليه،ولأنه يُحتمل وجود أكثر من شركة فى قاعدة البيانات بهذا الإسم لهذا سيكون الـ key ليس اسم الشركة ولكن سنستخدم ShipperId كمفتاح للتعديل على الصف.

مثال لتحديث بيانات صف فى الجدول:

افتح نافذة استعلام جديدة ، وتأكد أنك علي قاعدة البيانات Northwind ثم ادخل الاستعلام التالي:

```
Update  shippers
Set companyname='EgyptSoftware'
Where shipperid=4
```

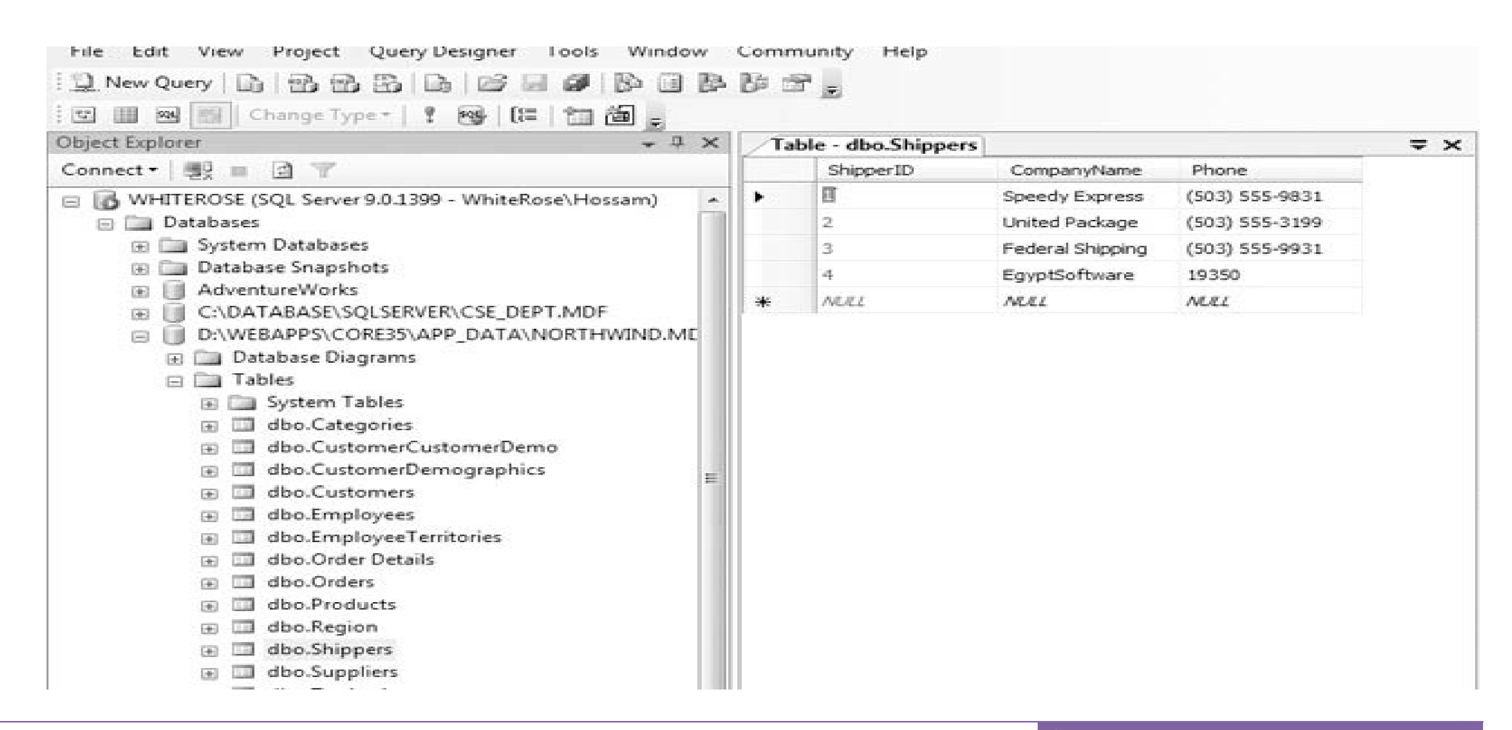
لتوضيح ماحدث فإننا استخدمنا shipperId كمفتاح للجدول وبالتالي سيسهل الوصول الى اى عنصر فى أى صف كما فى مثالنا وتلاحظ بعد ضغطك F5 ظهور النتيجة `(1 row(s) affected)`ولرؤية اثر التحديث كل ماعليك هو ان تفتح التسلسل الشجري لقاعدة البيانات على اليسار من Object Explorer وتفتح الجدول shippers بالضغط كلك يمين وتختار Open Table لتشاهد كما بالشكل :



كما لاحظت فقد تم تغيير اسم الشركة بنجاح ،لكن ماذا عن تحديث أكثر من عمود ؟
بلا شك الأمر بسيط سنستخدم الكلمة SET مرة واحدة ثم سنستخدم الفاصلة لتحديد أى من الأعمدة نريد تغيير ها فى اى صف، كمثال على ذلك لنفترض تغيير اسم الشركة بجانب هاتف الشركة:

```
Update shippers
Set copmanyname = 'EgyptSoftware',
Phone='19350'
Where shipperid = 4
```

لتشاهد التحديث على العمودين كما بالشكل :



حذف البيانات Deleting Data

من الأمور الهامة التي يجب أن تعرفها فى التعامل مع البيانات هى حذف البيانات.ونقوم بهذا الأمر باستخدام الجملة DELETE. وكما نوهنا فى الجملة UPDATE من توخى الحذر مع جملة الشرط WHERE فينبغي أيضا توخي الحذر مع الجملة DELETE، فمن السهل حذف كل الصفوف اذا لم نحسن التعامل مع جملة الشرط.

تقوم جملة DELETE بحذف الصف بكامله بغض النظر عن تحديد اى قيم فيه، وتتكون جملة الحذف من التالي:

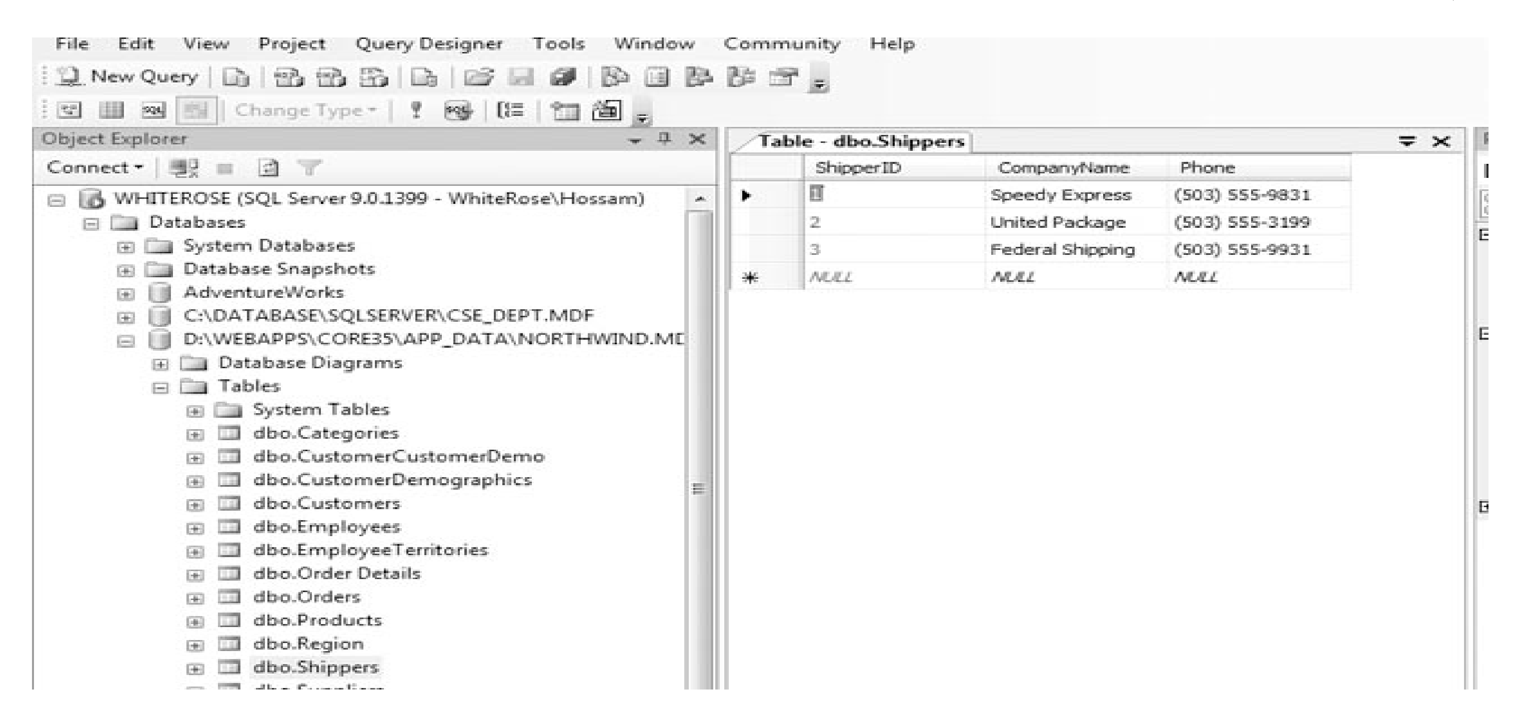
```
DELETE FROM <table>
WHERE <predicate>
```

جملة where اختيارية ولكن نكرر التحذير أنك دونها ستحذف كل شئ.

اذا أردت أن تحذف بعض السجلات من الجدول shippers فعليك تحديد المفتاح الرئيسي أولا لكل صف تريد حذفه، ومثال على ذلك قم بكتابة هذا الاستعلام واضغط f5 :

```
Delete from shippers
Where shipperid = 4
```

من المؤكد انه سيظهر لك هذه الرسالة `(1 row(s) affected)`، قم كما فى السابق بتصفح الجدول shippers لترى النتيجة كما بالشكل :



تم حذف الصف الأخير الذي يحوى اسم الشركة وهاتفها كما ترى.

لا تحاول حذف اى من العناصر الثلاثة فى الجدول (لأن هذه العناصر مرتبطة كمفتاح ثانوي F.K مع الجدول Orders) وسينتج database error لأن SSE يمنع حذف العناصر فى جدول مرتبطة بجدول آخر(يُمكنك الاستزادة بالرجوع مرة اخري لفهم المفاتيح فى الفصل الأول).

ماذا عن حذف كافة البيانات دفعة واحدة بدلا من مجموعة سجلات ؟ ، سنستخدم الجملة TRUNCATE TABLE فقط تعطها اسم الجدول وهى ستتولى المهمة ، وهى أفضل من جملة DELETE لأنها لاتقوم بعمل logging وتعني حفظ كل صف من الجدول فى log file قبل حذفه فى حين أن الجملة DELETE تقوم بذلك.

خاتمة الفصل

فى هذا الفصل تعلمنا كيفية التعامل مع البيانات وكيفية اضافة وحذف وتحديث هذه البيانات وكيفية المقارنة بين البيانات واستخدام المعاملات Operators .

الفصل الرابع

استخدام الإجراءات المُخزنة Using Stored Procedures

ماهي Stored Procedures

تعتبر الإجراءات المُخزنة Stored Procedures والتي سنشير لها فى الكتاب بـ (SP) جمل SQL تُمكنك من عمل إجراءاتك على قاعدة البيانات بصورة متكررة.
بإمكانك انشاء procedure مرة واحدة ثم تعيد استخدامه أكثر من مرة فى البرنامج الذي تعمل عليه.
لماذا كل هذا ؟ .. هذه الطريقة ستريحك كثيراً حال وجود اخطاء فى الكود المكتوب وعمل اصلاح لهذا الخطأ ، كما أنها تمكن تطبيقاتك من الولوج الى قاعدة بياناتك بطريقة سلسة وبكفاءة.
في هذا الفصل ، سيُصبح عندك المام جيد بالـ SP وكذا كيفية استخدامه داخل أكواد برامجك بالـ C#

في هذا الفصل سنتعلم :

- كيفية انشاء SP
- كيفية التعديل على SP
- استعراض SP definitions
- اعادة تسمية SP
- كيفية استعمال SP داخل أكواد C#
- حذف SP

كيفية انشاء Stored Procedures

يُمكن للـ SP أن تحوي المعاملات *Parameters* والتي تعني أننا يمكن ان نعيد قيم مدخلات ومخرجات ، حتي القيمة الافتراضية صفر ، ويمكن ان تعيد هذه المعاملات اكثر من قيمة او لاشئ. *(فكرة المعاملات مشتهرة فى عالم البرمجيات وهي تعتبر متغيرات لكن لإرجاع قيم للدوال)*
يُمكن استدعاء SP من داخل أكواد برمجية كالـ C# مثلاً، وأيضا يُمكن استدعاؤها من داخل SP آخر. بسبب قوة وسلاسة SP أصبحت هي الصيغة الأكثر انتشاراً في عالم برمجة قواعد البيانات، خاصة فى البرمجيات المتقدمة التي تستخدم الطبقات المتعددة Multitier Applications وكذلك في الـ Web Service ما فيه من تداخل بين قطاعات متعددة من البنية التحتية كالشبكات والأجهزة الخادمة Servers ، لأن لها القدرة على التخفيف من حدة ضغط الطلبات Requests على هذه الخوادم وعلى الشبكة.

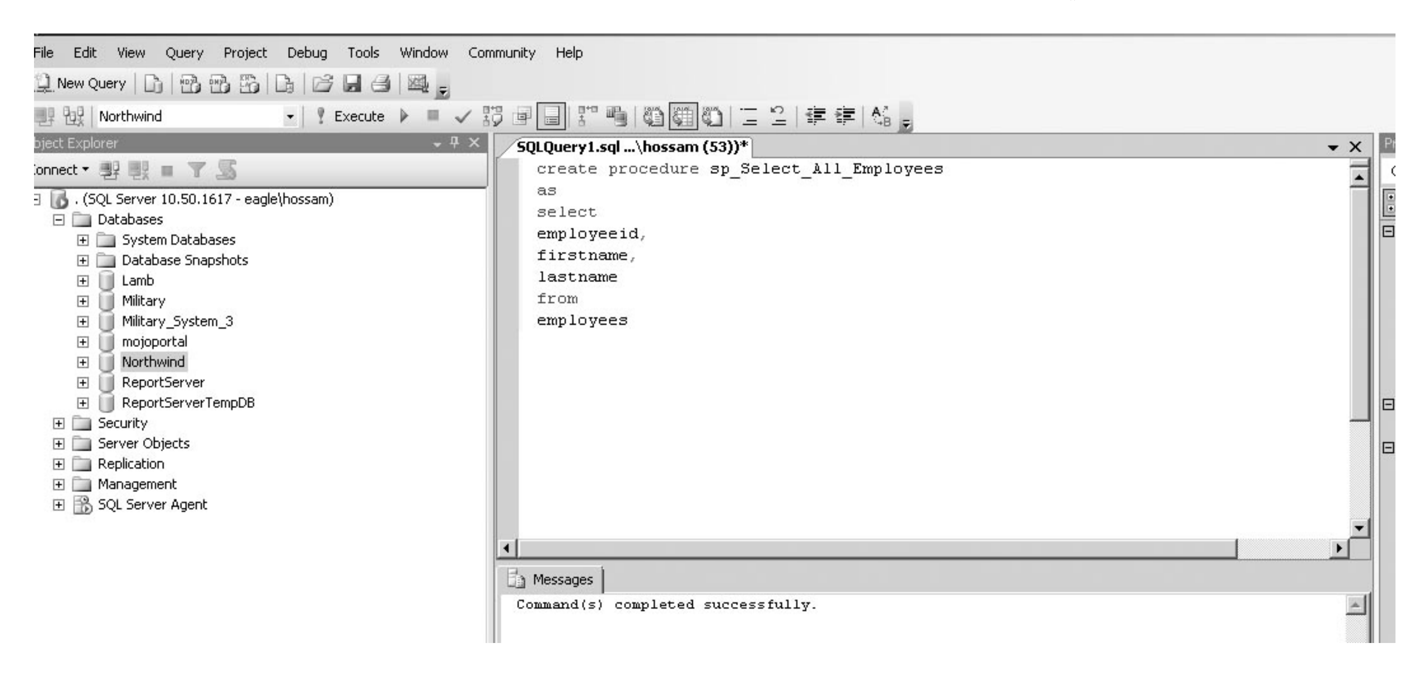
كيفية العمل على Stored Procedure مع SQL Server

سنقوم الآن بإنشاء أول SP لنا والذي يقوم بسرد أسماء العاملين فى قاعدة البيانات التي نعمل عليها Northwind ، وبالتالي لن نحتاج الي عمليه ادخال في SP.

١. افتح SSMS ثم قم بالاتصال بقاعدة البيانات *(لاحظ هنا اننا نستخدم SQL SERVER Management Studio النسخة الكاملة وليس Express) وفي كلتا الحالتين فإن الاجراء سينجح كما ذكرنا في أول الكتاب فى التبديل بين النسختين لن تجد أي فارق.*
٢. قم بتوسعة شجرة قواعد البيانات من اليسار ثم اختر قاعدة البيانات Northwind ثم كلك يمين واختر New Query ثم ادخل الاجراء التالي واضغط F5 :

```
create procedure sp_Select_All_Employees
as
  select
      employeeid,
      firstname,
      lastname
from
  employees
```

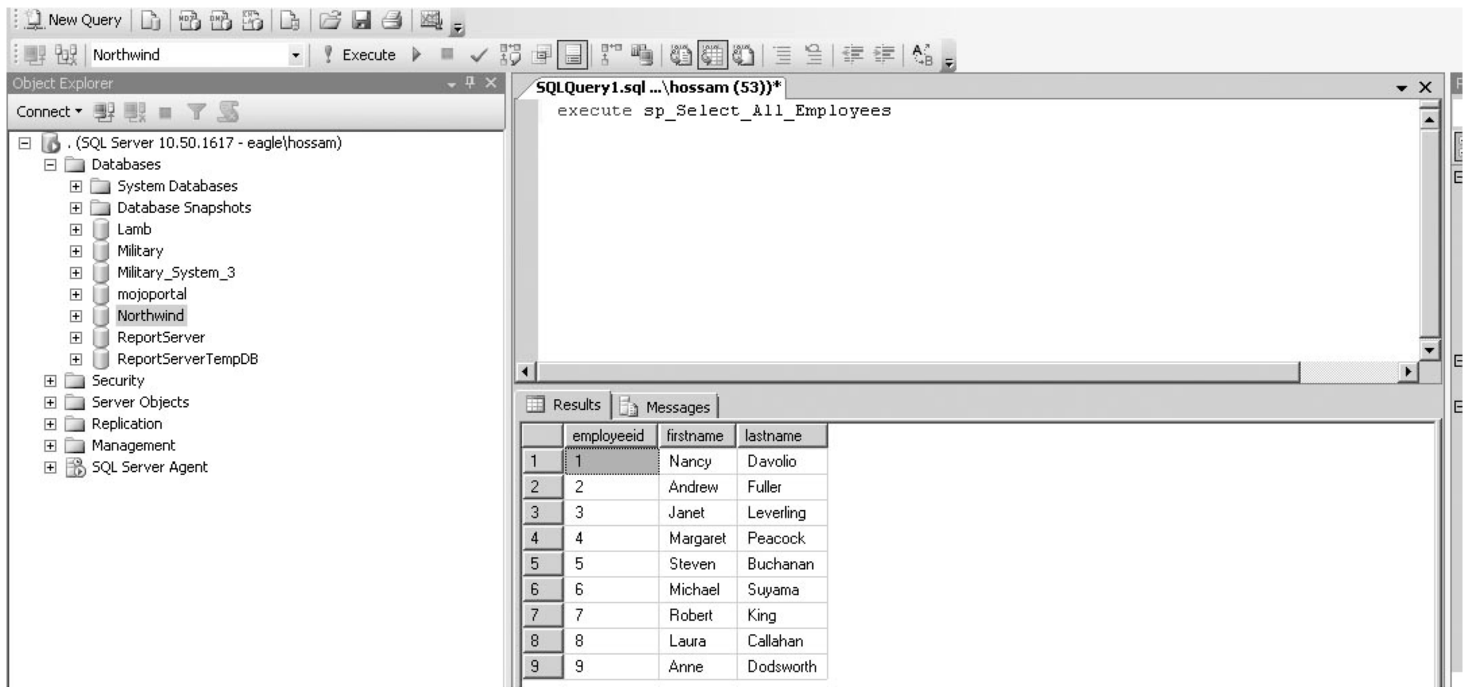
لترى النتيجة كما بالشكل التالي:



٣. ولتنفيذ هذا الإجراء ، أدخل الاستعلام التالي ، ثم اضغط F5 :

```
execute sp_Select_All_Employees
```

لترى النتيجة كما بالشكل:



لتوضيح ماحدث:
قمنا بانشاء SP عن طريق الجملة Create Procedure ، وقمنا بالفصل مابين اسم الاجراء والمعاملات (والتي لم نستخدمها فى مثالنا) بالكلمة المحفوظة AS والتي تُسمى Signature وما بين الإجراء الفعلي (جمل SQL).
وبعد الكلمة AS تم وضع الجملة الاستعلامية

```
Select
  employeeid,
  firstname,
  lastname
from
   employees
```

بعد أن قام SSMS بإنشاء الإجراء لنا ، قمنا بعمل تنفيذ لهذا الاجراء عن طريق الأمر:

```
execute sp_Select_All_Employees
```

هذا كل شئ ، يتوقف صعوبة الـ SP على تعقيدات جمل SQL التي يتضمنها برنامجك ، وبالتالي SP لا يُعد صعبا في حد ذاته.

تنويه:
<u>لعلك لاحظت اسم الاجراء قد بدأ بالحرفين sp_ ، وجرت العادة على تسمية الاجراءات المنشئة بواسطة T-SQL مبتدأة بـ sp ، وتسمية الإجراءات المُنشئة خارج T-SQL بـ xp_ وتعني extended procedure ، وليست هذه قاعدة ، فقد نجد مئات من الاجراءات مبتدأة بـ sp لكنها في الحقيقة ليست مكتوبة بـ T-SQL.</u>

لماذا التنوية؟
في الحقيقة لايُنصح باستعمال sp_ في مقدمة اسم الإجراء المزمع انشاؤه وذلك حتي تكون عملية البحث عن الاجراء اكثر سرعة ، لأنني نوهت ان أغلب الاجراءات المكتوبة لـ T-SQL تبدأ بـ sp وبالتالي سيقوم محرك قاعدة البيانات بالمرور على هذه الاجراءات واحداً تلو الآخر حتى يصل للإجراء المطلوب ، فعند البحث عن أي اجراء يحدث هذا السيناريو:

- يقوم SQL Server بالبحث داخل قاعدة البيانات Master ابتداءاً عن هذا الإجراء ، ومن المعروف أنها تحوي مئات الاجراءات مبتدأة بـ sp_ ، فإذا وجد الإجراء المطلوب فإنه سوف يقوم بالمناداه عليه calling.
- اذا كان الإجراء غير موجود في قاعدة البيانات master وهو الغالب ، فإنه يقوم بالبحث عن هذا الاجراء داخل قاعدة البيانات الحالية (database_name.stored_procedure_name) .

يعني فى النهاية سيمر على قاعدة بيانات غير معنية بالبحث حتى ولو قمت بتوفير اسم قاعدة البيانات الحاوية للإجراء!
كما ينبغي التنوية بعدم استخدام اسم اجراء موجود في قاعدة البيانات الرئيسية master ، صدقني حتى ولو وفرت اسم قاعدة البيانات التي انشأتها بنفس اسم الاجراء سينادي الاجراء الرئيسي في القاعدة master.

إنشاء Stored Procedure يحوي معاملات

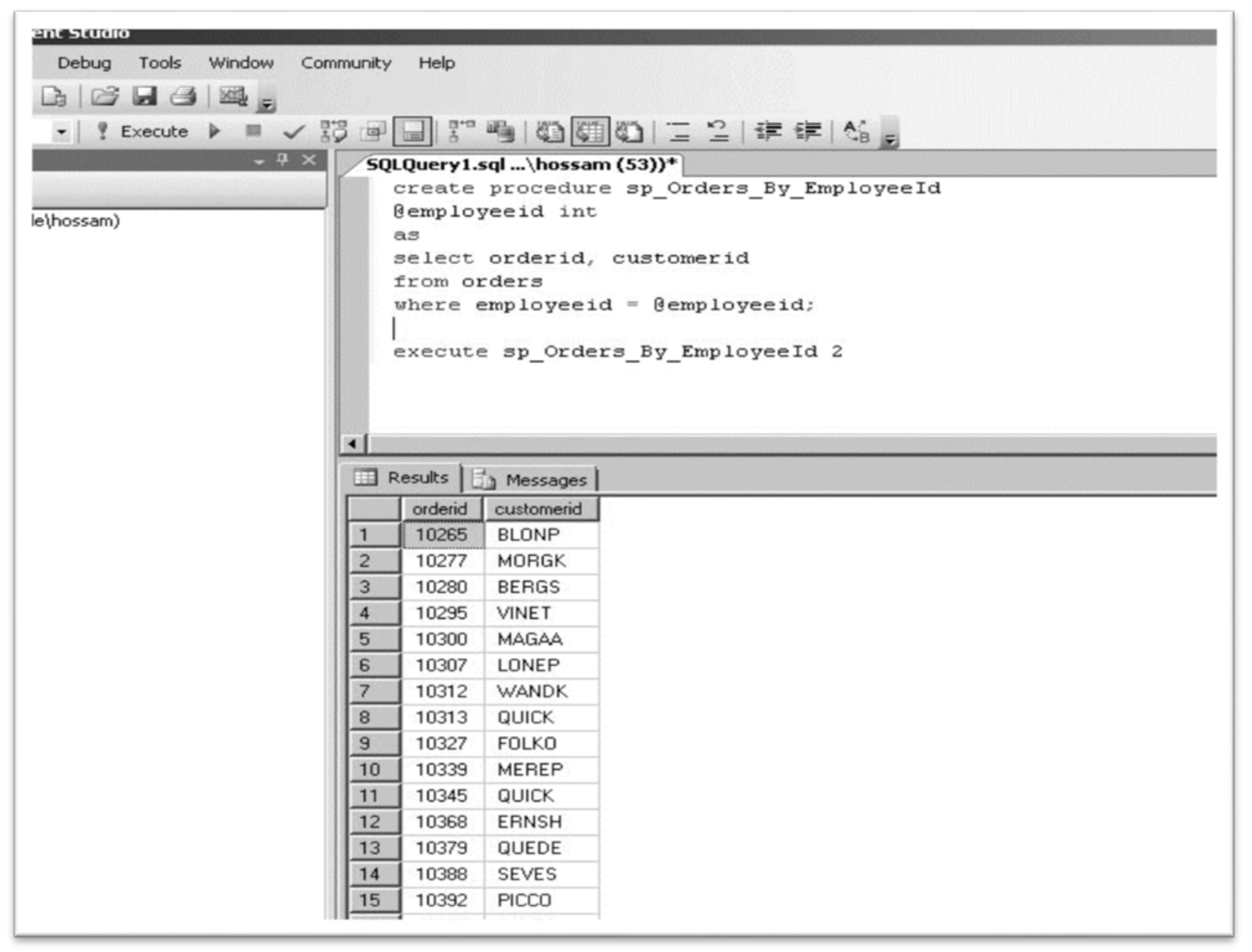
سنقوم بإنشاء SP يقوم بعرض عدة طلبيات لموظف ما، سنقوم بتمرير مُعرف الموظف ID الى SP ليتم استخدامه في الاستعلام.

١- قم بادخال الاستعلام التالي ، واضغط F5 ، لترى رسالة " Command(s) completed successfully" في نافذة النتائج.

```
create procedure sp_Orders_By_EmployeeId
@employeeid int
as
select orderid, customerid
from orders
where employeeid = @employeeid;
```

٢- ولتقوم بتنفيذ الاجراء السابق ، قم بادخال الأمر التالي ، مع قيمة المعامل" ٢ " لترى نتيجة كما بالشكل :

```
execute sp_Orders_By_EmployeeId 2
```



لعلك لاحظت أننا قمنا بعمل تنفيذ أكثر من استعلام في نفس النافذة ، وهما تعليمتي الانشاء والاستدعاء للإجراء السابق ، وهذا ما يتميز به SQL Server عن باقي أنظمة ادارة قواعد البيانات ، كما ويمكنك أن تظلل استعلاماً ما ليتم تنفيذه بصفة مستقلة.
ولتوضيح ما حدث ، فالأمر كما تعلمت مسبقاً ، اذا قمنا بانشاء اجراء مُخزن SP عن طريق الأمر CREATE PROCEDURE ثم نضع اسم المعامل المرغوب كما في مثالنا ، @employeeid int قبل الكلمة المحفوظة AS واما الكلمة int فهي نوع المعامل data type ، ثم قمنا بوضع الشرط where ، والذي فيه نحدد ان يكون اسم العنصر في العمود employeeid مساوٍ للمعامل @employeeid

```
where
employeeid = @employeeid;
```

هذا كان مثال بإدخال معامل ، ماذا عن استخدام معامل في الإخراج ؟
معاملات الاخراج Output Parameters تُستخدم في تبادل القيم مابين الاجراءات المُخزنة SPs ، ولكن أحياناً تُستخدم في الأكواد البرمجية مثل C# ، لذا هذا المثال سيكون في كيفية انشاء معاملات اخراج يتم استخدامها فيما بعد في برامجنا ، وسوف نعرف كيف نسترجع قيمة بخلاف القيمة "صفر".
١- ادخل الاستعلام التالي ، ثم اضغط F5 للتنفيذ ، لترى الرسالة " Command(s) completed successfully " في نافذة النتائج.

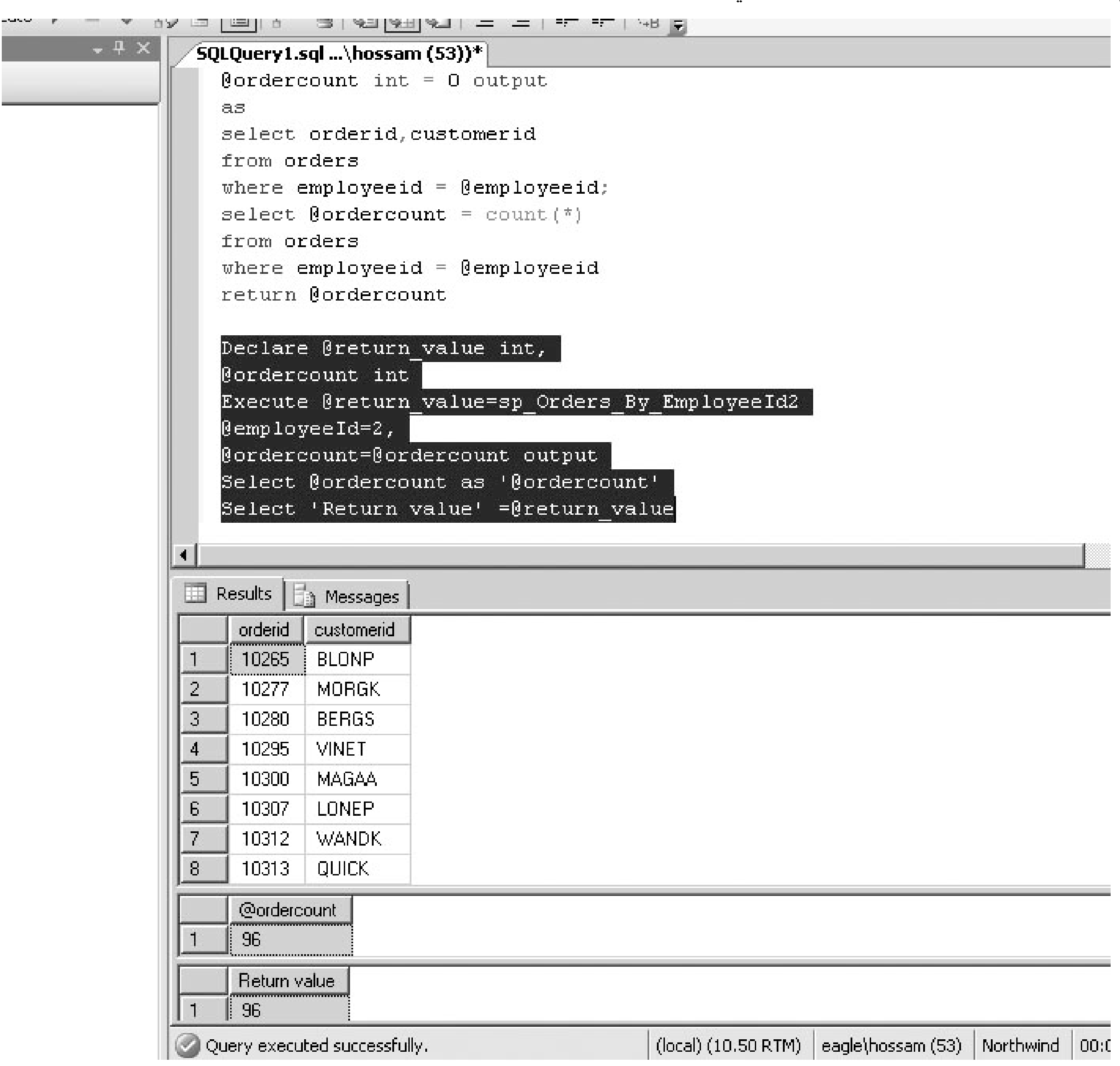
```
Create procedure sp_Orders_By_EmployeeId2
@employeeid int,
@ordercount int = 0 output
as
select orderid,customerid
from orders
where employeeid = @employeeid;
select @ordercount = count(*)
```

from orders
where employeeid = @employeeid
return @ordercount

٢- يُمكنك الان اختبار الإجراء السابق ، فقط أدخل الاجراء التالي في نافذة الاستعلامات ، لكن تنبه أنك قد قمت بمسح الاستعلام السابق ، او قم بتظليل الاستعلام التالي ، حتى لايفهم نظام قواعد البيانات انك تُنشئ اجراء مُخزن مرتين:

Declare @return_value int,
@ordercount int
Execute @return_value=sp_Orders_By_EmployeeId2
@employeeId=2,
@ordercount=@ordercount output
Select @ordercount as '@ordercount'
Select 'Return value' =@return_value

ثم اضغط F5 للتنفيذ ، لترى كما بالشكل التالي:



كما ترى فإن النتيجة المرجوة هي "96".
ولتوضيح ماحدث ، فإننا أضفنا المعامل @ordercount ثم قمنا باسناد القيمة "0" له

Create procedure sp_Orders_By_EmployeeId2
@employeeid int,
@ordercount int = 0 output

```
as
select orderid,customerid
from orders
where employeeid = @employeeid;
```

واما الكلمة output فهي من الكلمات المحفوظة والتى توضح أن هذا المعامل لإخراج القيم ، لاحظ معي الفاصلة المنقوطة التي أنهينا بها الاستعلام أعلى (;) ـ كما قمنا بإضافة الاستعلام التالي :

```
select @ordercount = count(*)
from orders
where employeeid = @employeeid
```

كما نوهنا على الفاصلة المنقوطة والتي يبرز أهميتها هنا لأنها تفصل مابين الاستعلامين السابقين ، واسندنا الدالة count(*) الى المعامل @ordercount ثم استرجعا القيمة باستخدام الكلمة المحفوظة return

```
return @ordercount
```

ومن الملاحظ انه دائما ماتعود الدالة COUNT برقم صحيح INTEGER ،لذا فأنت مسبقاً تعرف أن الدالة RETURN ستعود برقم وهو "96" كما في مثالنا.

بالقطع هناك طرق عدة لكتابة مثل هذا المثال ، لكن هدفنا هو تعليم الطريقة وليس الفكرة ، والأهم هو كيفية استخدام هذه الأمثلة مع أكواد C# ،ومع هذا سنُكمل طريقنا في هذا الفصل بكيفية التعديل وحذف الاجراء المُخزن.

تعديل الـ Stored Procedure

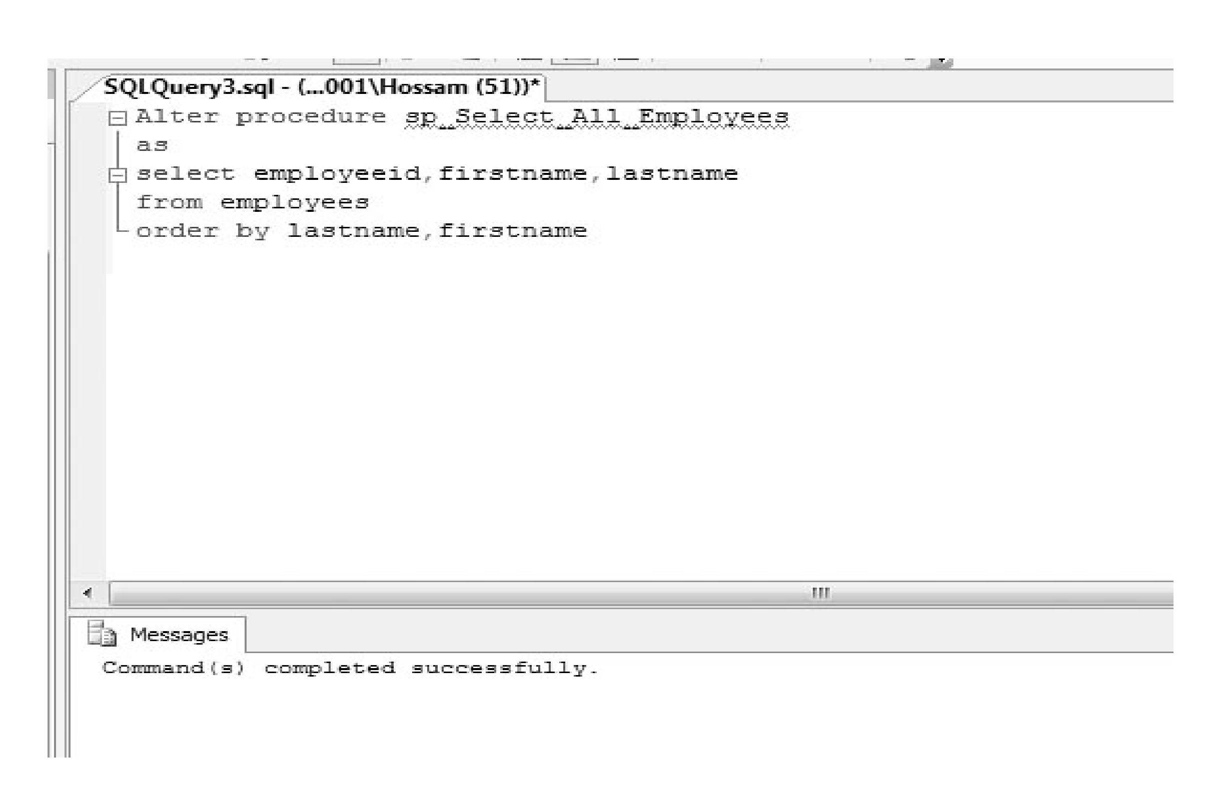
سنتطرق الآن لكيفية اجراء تعديلات على الـ SP sp_select_All_employees الذي أنشأناه

اتبع الخطوات التالية:

أدخل الاستعلام التالي ثم F5 :

```
Alter procedure sp_Select_All_Employees
as
select employeeid,firstname,lastname
from employees
order by lastname,firstname
```

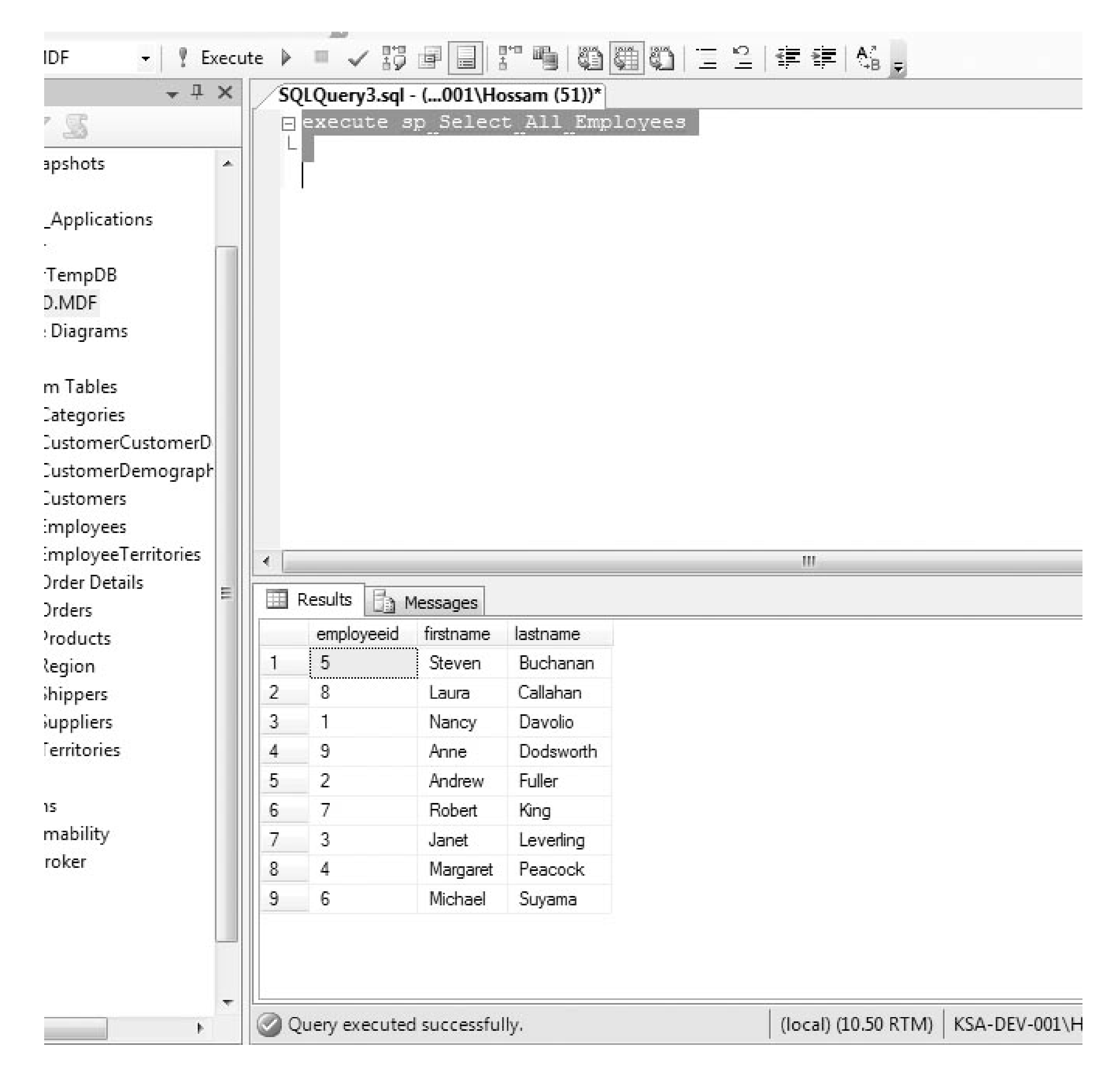
لترى كما بالشكل:



بعد ذلك قم بعمل تنفيذ للإجراء execute عن طريق الاستعلام التالي:

execute sp_Select_All_Employees

وقارنها بالنتيجة التي ظهرت سابقاً لترى العمود الجديد الذي تمت اضافته عن طريق التعديل كما بالشكل:



لفهم ماحدث ،فإننا قمنا بإضافة الاستعلام ALTER PROCEDURE ملحوقاً باسم الإجراء المراد تعديله ، ليقوم بعمل تحديث له

Alter procedure sp_Select_All_Employees

كما قمنا بعمل ترتيب للعناصر باستخدام OREDER BY تصاعديا حسب الإسم الأخير last name ثم الإسم first name .

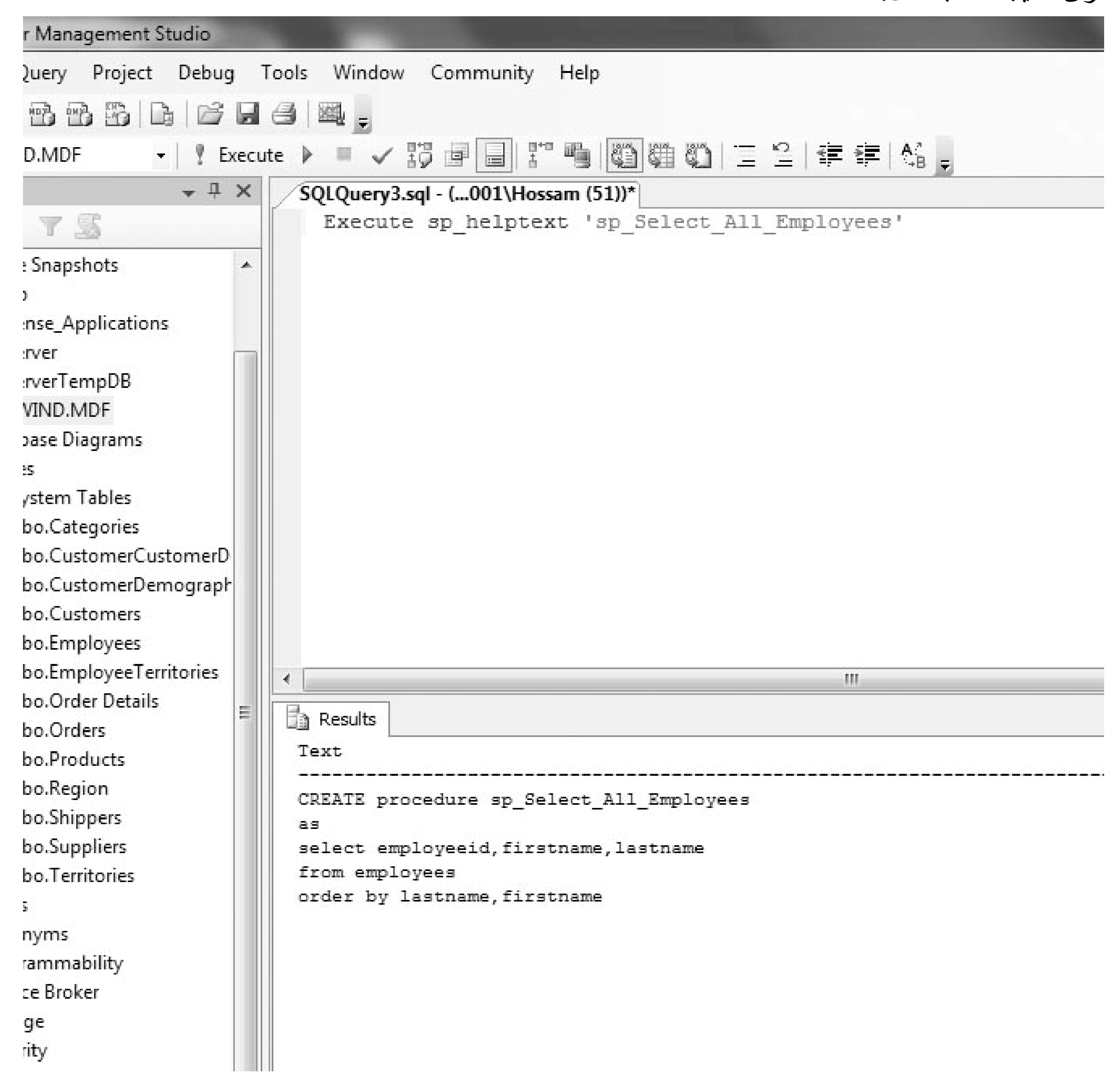
order by lastname,firstname

استعراض SP Definitions

يوفر SQL SERVER آلية لاستعراض التوصيف الخاص بالعناصر المنشئة في قاعدة البيانات Definition of Objects.
ويعرف هذا بعملية *metadata retrieval*. تُخزن كافة المعلومات المتعلقة بعناصر قواعد البيانات في إجراءات مُخزنة تابعة لنظام قواعد البيانات ، والتي تُمكنك من استرجاعها وقتما تشاء.
لنقوم بتجربة مثال على ذلك:
لكي تستعرض توصيف الإجراء السابق sp_Select_All_Employees قم بكتابة الاستعلام التالي :

```
Execute sp_helptext 'sp_Select_All_Employees'
```

قبل أن تنفذ الاستعلام ، اذهب الى القائمة query ثم اختر Result to -> Result to text
لترى النتيجة كما بالشكل:



لفهم الاستعلام السابق ، فإن السر يكمن في الاجراء(sp_helptext) المُخزن مسبقاً فى إجراءات نظام قواعد البيانات Predefined SQL Server SP والتي تأخذ اسم العنصر (الإجراء) المراد اظهار الـ definition له. هذا الإجراء المُخزن sp_helptext لا يعمل على الجداول فلا تستطيع استعراض عناصر جدول عن طريقه.

إعادة تسمية SP

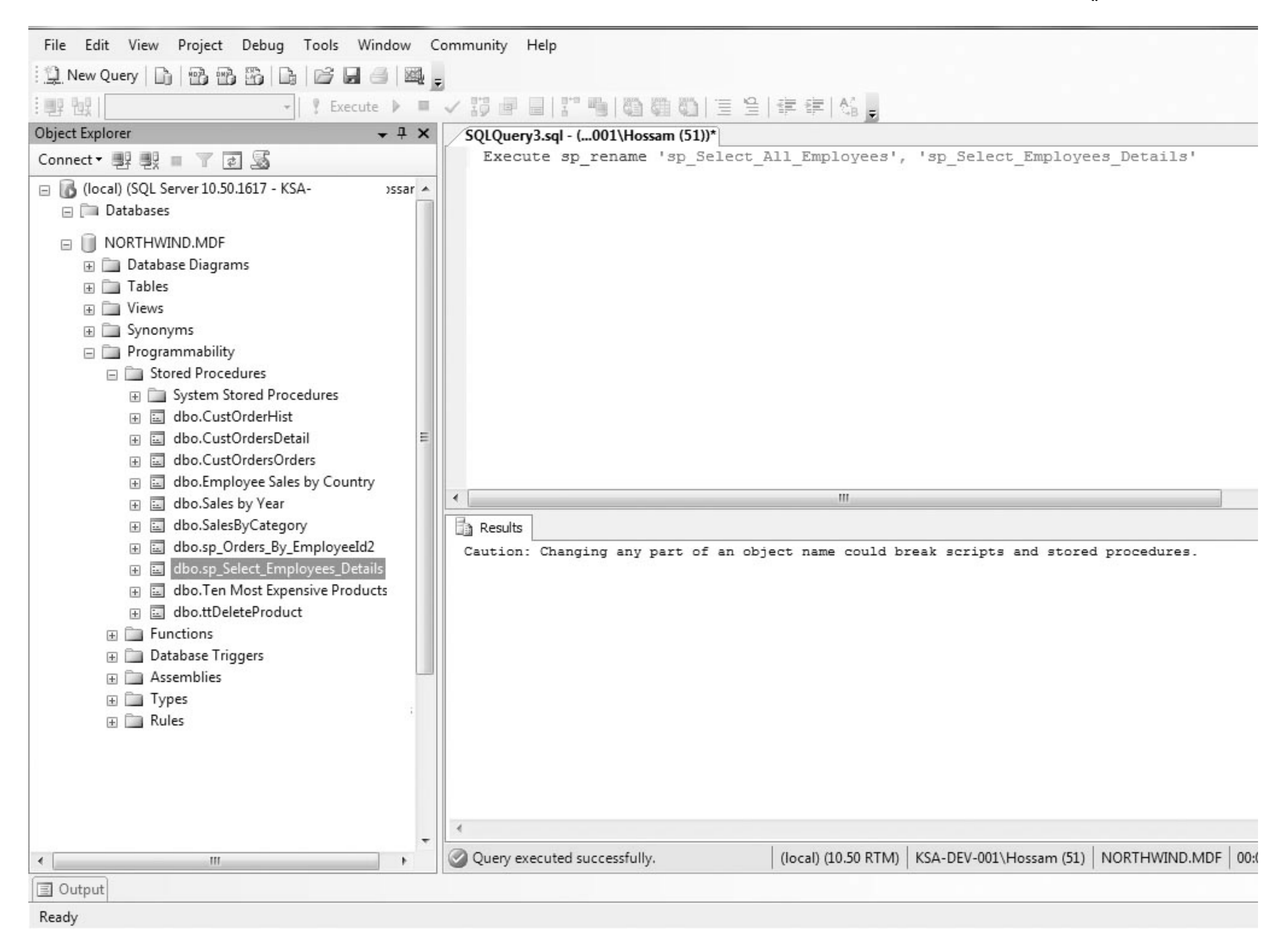
يُمكنك اعادة تسمية اجراءاتك عن طريق استخدام الإجراء المُعرف مسبقاً sp_rename لتتعرف على ذلك تابع المثال التالي:
أدخل الاستعلام التالي فى نافذة الاستعلامات :

Execute sp_rename 'sp_Select_All_Employees', 'sp_Select_Employees_Details'

ثم اضغط Execute لترى الرسالة التالية في نافذة النتائج :

"Caution: Changing any part of
an object name could break scripts and stored procedures."

على الرغم من نجاح تنفيذ الاستعلام السابق ! ، ولتتأكد من ذلك قم بعمل استعراض للإجراء الذي أنشأناه عن طريق توسعة شجرة قاعدة البيانات Northwind ثم توسعة الفرع Programmability ثم Stored Procedures ثم كلك يمن واختر referesh ستلاحظ أن الإجراء السابق المسمى sp_Select_All_Employees قد تغير اسمه الى sp_Select_Employees_Details كما بالشكل التالي:



قام الإجراء المخزن مسبقا sp_rename بتغيير اسم العنصر القديم عن طريق اسناد اسم جديد كما رأينا فى الاجراء السابق ، ومن مزايا هذا الإجراء انه يعمل مع كافة العناصر سواءا أكانت جداول أو أعمدة ..الخ.

كيفية استعمال SP داخل أكواد C#

حان الوقت لكتابة أكواد C# تمكنك من التعامل مع إجراءاتك السابق عملها ، سنقوم الآن بعمل execute للإجراء sp_Select_Employees_Details .

- قم بفتح مشروع جديد من النوع Consol Application وقم بتسميته CallSp1
- قم بتسمية الملف Program.cs الى CallSp1.cs

قم بنسخ الكود التالي فى الملف CallSp1.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
namespace CallSp1
{
    class CallSp1
    {
        static void Main()
        {
            // create connection
            SqlConnection conn = new SqlConnection("server=.;integrated security = true;database =
 northwind.mdf");
            try
            {
                // open connection
                conn.Open();
                // create command
                SqlCommand cmd = conn.CreateCommand();
                // specify stored procedure to execute
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.CommandText = "sp_select_employees_details";
                // execute command
                SqlDataReader rdr = cmd.ExecuteReader();
                // process the result set
                while (rdr.Read())
                {
                    Console.WriteLine(
                    "{0} {1} {2}"
                    , rdr[0].ToString().PadRight(5)
                    , rdr[1].ToString()
                    , rdr[2].ToString());
                }
                rdr.Close();
            }
            catch (SqlException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            finally
            {
                conn.Close();
            }
        }
    }
}
```

قم بتنفيذ الكود السابق بالضغط على Ctrl+F5 لترى النتيجة كما بالشكل:

```
C:\Windows\system32\cmd.exe
5     Steven Buchanan
8     Laura Callahan
1     Nancy Davolio
9     Anne Dodsworth
2     Andrew Fuller
7     Robert King
3     Janet Leverling
4     Margaret Peacock
6     Michael Suyama
Press any key to continue . . .
```

لشرح الكود بأعلاه لاحظ التعليق create command // وأسفله الكود المسئول عن انشاء الأمر الذي من خلاله سنستدعي الإجراء الذي انشأناه في قواعد البيانات sp_select_employees_details لهذا قمنا بتخصيص command.type =CommandType.StoredProcedure وقمنا باسناد اسم الإجراء الى cmd.CommandText، وبعدها قمنا بتنفيذ هذا الإجراء عن طريق الأمر SqlDataReader rdr =cmd.ExecuteReader(); ، أما باقي الكود فهو لتنظيم عرض ناتج الإجراء السابق.

لنجرب مثال آخر على كيفية استدعاء إجراء يحتوي معاملات parameters :

- قم بفتح مشروع جديد من النوع Consol Application وقم بتسميته CallSp2
- قم بتسمية الملف Program.cs الى CallSp2.cs
- قم بنسخ الكود التالي إلى الملف CallSp2.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Text;

namespace Callsp2
{
    class Program
    {
        static void Main(string[] args)
        {
            // create connection
            SqlConnection conn = new SqlConnection(@"
            server = .;
            integrated security = true;
            database = northwind.mdf
            ");
            try
            {
            // open connection
            conn.Open();
            // create command
            SqlCommand cmd = conn.CreateCommand();
            // specify stored procedure to execute
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.CommandText = "sp_orders_by_employeeid2";
            // create input parameter
            SqlParameter inparm = cmd.Parameters.Add(
            "@employeeid", SqlDbType.Int);
            inparm.Direction = ParameterDirection.Input;
            inparm.Value = 2;
            // create output parameter
            SqlParameter ouparm = cmd.Parameters.Add(
            "@ordercount", SqlDbType.Int);
            ouparm.Direction = ParameterDirection.Output;
            // create return value parameter
            SqlParameter retval = cmd.Parameters.Add(
            "return_value", SqlDbType.Int);
            retval.Direction = ParameterDirection.ReturnValue;
            // execute command
            SqlDataReader rdr = cmd.ExecuteReader();
            // process the result set
            while (rdr.Read())
            {
                Console.WriteLine(
                "{0} {1}"
                , rdr[0].ToString().PadRight(5)
                , rdr[1].ToString());
```

```
            }
            rdr.Close();
            // display output parameter value
            Console.WriteLine(
            "The output parameter value is {0}"
            , cmd.Parameters["@ordercount"].Value);
            // display return value
            Console.WriteLine(
            "The return value is {0}"
            , cmd.Parameters["return_value"].Value);
            }
            catch (SqlException ex)
            {
                Console.WriteLine(ex.ToString());
            }
            finally
            {
                conn.Close();
            }
        }
    }
}
```

- قم بتنفيذ الكود بالضغط على ctrl+ F5 لتشاهد كما بالشكل التالي:

```
C:\Windows\system32\cmd.exe
10990 ERNSH
10994 VAFFE
11000 RATTC
11001 FOLKO
11005 WILMK
11009 GODOS
11010 REGGC
11013 ROMEY
11014 LINOD
11015 SANTG
11020 OTTIK
11028 KOENE
11032 WHITC
11035 SUPRD
11042 COMMI
11053 PICCO
11059 RICAR
11060 FRANS
11070 LEHMS
11073 PERIC
The output parameter value is 96
The return value is 96
Press any key to continue . . .
```

كما بالكود السابق ،قمنا بإنشاء الأمر الذي يستدعي الاجراء من قاعدة البيانات ، وقمنا بتنفيذه بالآلية السابقة ، الذي يختلف هو إضافة الأسطر الخاصة بالمعاملات والتي ستجدها أسفل التعليق create input parameter // مع تخصيص طبيعتها من حيث الإدخال أو الإخراج Output/ ParameterDirection.Input أو ParameterDirection.ReturnValue ، ثم قمنا باسناد القيمة (2) لمعامل الإدخالك inparm.value = 2 ; ، وقمنا بعرض القيم الناتجة لمعامل الإخراج عن طريق الكود التالي:

```
// display output parameter value
Console.WriteLine(
"The output parameter value is {0}"
, cmd.Parameters["@ordercount"].Value);
```

```
// display return value
Console.WriteLine(
"The return value is {0}"
, cmd.Parameters["return_value"].Value
);
```

بإمكانك أن تُنشئ أي عدد من معاملات الإدخال والإخراج كما تشاء ، كما انوه على حتمية اسناد معاملات الكود command parameters إلى معاملات الادخال التي لا تحوي قيم افتراضية default values.

كما انوه على انه ليس عليك أن تسند معاملات الكود إلى أي معاملات إخراج لن تستخدمها ، كما يجب أن يكون اسم المعامل في الكود مناظر لإسم المعامل فى قاعدة البيانات سواءاً كان إدخال او اخراج (لا تعر بالاً اذا كان هناك أحرف كبيرة capital أو صغير small في اسم المعامل لأن T-SQL ليست حساسة لحالة الأحرف Case Sensitive) ، أما القيمة المسندة للمعامل في الكود فأنت حر اختيار اي اسم تشاء لها.

على الرغم من قابلية معالجة معاملات الاسناد داخل كود ADO.NET، إلا أنه لابد من وجود قيمة عائدة return value واحدة حتمية. كما في معاملات الإخراج فإنك لست بحاجة لإنشاء معامل كود لإسترجاع قيمة مالم تنوي استخدامها بالفعل .

حذف الإجراء Deleting Stored Procedure

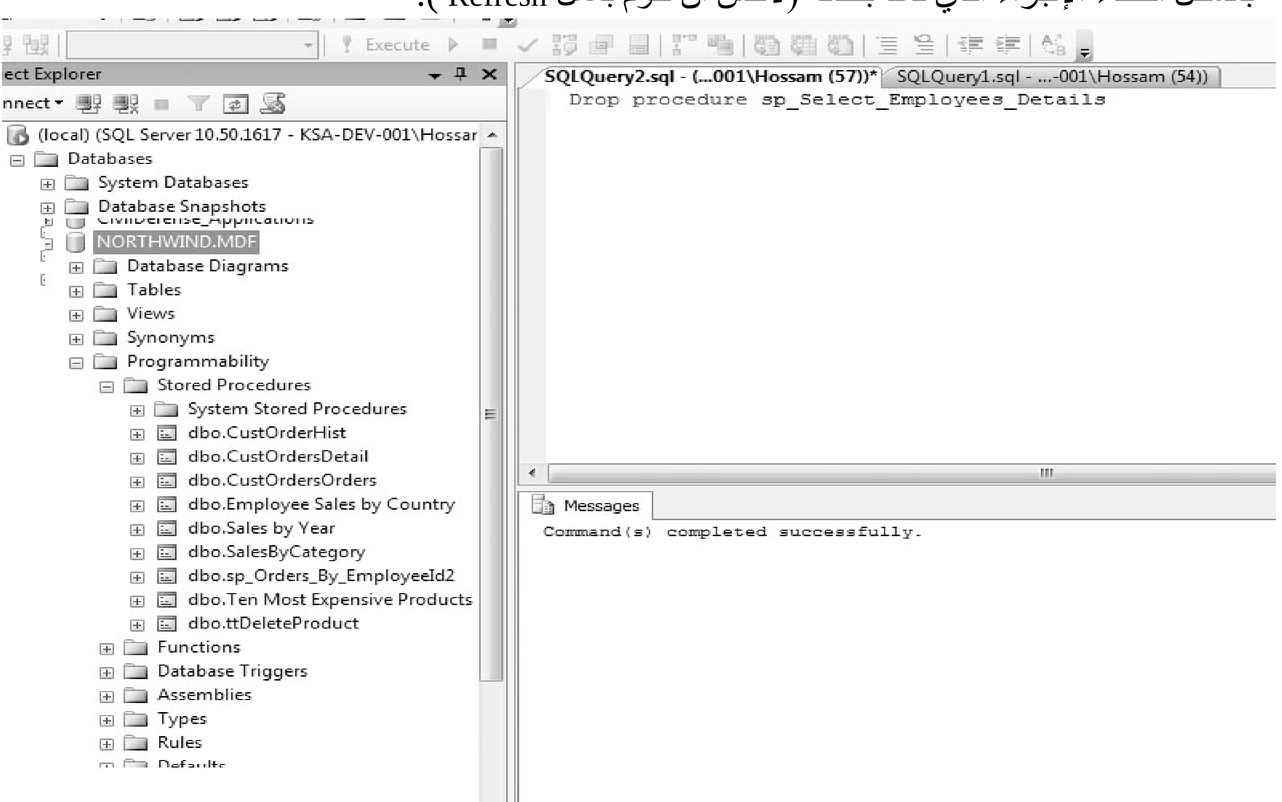
في حال استغناءك عن أي اجراء قمت بانشاءه مسبقاً يمكنك بكل بساطة حذفه ، سنقوم بحذف الإجراء الأول لنا sp_Select_All_employees الذي قمنا بتغيير اسمه إلى sp_Select_Employees_Details ولعمل ذلك قم الآتي:

- قم بكتابة الاستعلام التالي فى نافذة الاستعلامات في SQL Server

```
Drop procedure sp_Select_Employees_Details
```

لترى الرسالة التالية (Command(s) completed successfully).

- قم باستعراض قاعدة البيانات Northwind كما تعودت لتبحث عن الاجراءات الخاصة بقاعدة البيانات ، ستلاحظ كما بالشكل اختفاء الإجراء الذي قمنا بحذفه (لاتنس ان تقوم بعمل Refresh):



واضح أن الأمر DROP هو المسئول عن عملية حذف العناصر سواءاً كانت جداول او حتى إجراء كما في المثال السابق ، فلكي تحذف أى اجراء فقط اكتب الأمر DROP ملحوقاً باسم الإجراء المراد حذفه ، وبهكذا طريقة يتم حذفه.

خاتمة الفصل

تعلمنا في هذا لفصل كيفية انشاء الاجراءات ، كيفية استدعاءها وتنفيذها ، وكذا كيفية التعامل معها من خلال لغة البرمجة C# ومناداتها باستخدام ADO.NET، تعلمنا كيفية مناداة الإجراء Procedure Calling ، كيفية التعديل على الاجراء والحصول على المعلومات المخزنة عن الإجراء ، وكيفية التعامل مع الاجراءات التي تحوي معاملات Parameters ، وكذا إعادة تسميتها وحذفها ، في الفصل التالي، سنتعرف كيفية التعامل مع XML.

الفصل الخامس

استخدام XML

منذ 1996 وقبل الاصدار الأول لبيئة تطوير NET. ولغة XML ظهرت بعد عدة تجارب مخبرية ، ومن يومها وقد ازدادت الاعتمادية على هذه التقنية بشكل كبير في توصيف البيانات وكأداة رائعة لنقل البيانات من خلال الإنترنت ، وعليها قامت مايكروسوفت بإدراج امكانيات XML في كافة منتجاتها.
هدفنا في هذا الفصل تعريفك على هذه اللغة الرائعة في التعامل مع البيانات مع MS SQL Server 2008

في هذا الفصل سنتعلم الآتي:

- تعريف XML
- لماذا XML ؟
- مزايا تخزين البيانات على صورة XML
- تعرف على XML documents
- تعرف على التصريح XML Declaration
- تحويل البيانات العلائقية إلى XML
- كيفية تخزين واسترجاع XML documents باستخدام أنواع بيانات XML

تعريف XML

XML هي اختصار لـ eXtensible Markup Language وهي اشتقاق من اللغة القديمة SGML (Standard Generalized Markup Language) وهى تعتبر لغة توصيف *metalanguag* ،لغات التوصيف لا تُستخدم في البرمجة ، ولكن تُسند إلى لغات أخرى لتزيد من امكانياتها ، وهي لغة ترميز markup language، ولقد صممت لتحسين وظيفة الويب بتقديم وسيلة تمييز أكثر مرونة ومواءمة.
تتكون وثائق XML من اسطر نصوص مقروء بالعين وكذا مفهومة للحواسب ، كل وثيقة XML هو كيان لغوي مستقل مُعرف باستخدام عناصر(elements) حيث تقوم الكلمات المفتاحية للغة XML المعجم الداخلي Vocabulary لها بالتحقق من هذه الوثيقة من حيث بنية الجملة المكتوبة Syntax ودلالتها Semantics وتعتبر XML Schema Definition Language (XSD) هي المسئولة عن تحديد هذا المعجم.
في العام 1996 قامت منظمة الشبكة العنكبوتية (W3C) بتطوير لغة XML أملاً في دعم نطاق واسع من التطبيقات، ثم قامت باستخدامها في انشاء اللغة XHTML ومنذ ذلك الحين وهذه المنظمة مزمعة بانشاء عدة تقنيات ذات اعتمادية على XML بشكل واسع منها eXtensible Sylesheet Language (XSL) ، وهي لغة جيدة في اضفاء صفات جيدة لتصميم صفحات الويب كما في اللغة Cascading Style Sheets (CSS) المشهورة مع HTML ، وكذا طورت اللغة XSL Transformation (XSLT) ، والتي تقوم بالتحويل فيما بين وثائق XML المختلفة.

لماذا XML ؟

تعتبر XML لغة متعددة الأغراض ، واسعة الاستخدام في تقنيات تمثيل البيانات المختلفة وكذا التطبيقات المختلفة، كما أشرنا بأن قواعد البيانات العلائقية من ميزتها انها تتعامل مع البيانات مهيكلة structured فإن لغة XML تتميز بأنها تتعامل مع البيانات semi structured والغير مهيلكة بالمرة unstructured ، كما انها ذات تكامل تام مع نظام قواعد البيانات SQL Server 2008 وما قبله ،فتمكنك من استخراج البيانات والتعامل معها وكذا التحويل بينها وبين قواعد البيانات العلائقية.

مزايا تخزين البيانات على صورة XML

لا تعتمد XML على طبيعة النظام التي تعمل عليه platform-independent ، كما أنها توفر مرونة عالية لا تتوفر فى قواعد البيانات العلائقية في بعض حالات عرض البيانات ، إلا أننا يمكننا حصر هذه الميزات في التالي:

- بما أنها لغة ذاتية التوصيف self-describing فيمكن للتطبيقات التي تعتمد عليها أن تتضمن بيانات XML من معرفة البنية او التوصيف لهذه البيانات ، ويتم تضمينها في صورة شجرية ، تنقسم الى جذر Root وكذا فرع رئيسي Parent node وهذه الصورة تكون مايعرف بوثيقة XML.
- الطريقة التي تُكتب بها XML تحافظ على تنسيق محتواها لأنها دائما في صورة شجرية منظمة.
- يتم التحقق من صحة أي وثيقة XML عن طريق XML Schema.
- الطبيعة الشجرية المرتبة للغة XML ، تُمكنك من البحث بداخلها عن طريق استخدام أي من لغات الاستعلام والبحث داخل XML مثل Xquery و XPath.
- بما أنها لغة ممتدة extensible ، فمن السهل إجراء عمليات قواعد البيانات المعتادة عليها من اضافة وحذف وتعديل.

بالطبع هناك الكثير من المزايا ، كما أنوه على أنك كلما كتبت وثائق XML حسب الطرق القياسية المعتمدة من منظمة مثل W3C كلما استفدت بمزايا هذه التقنية.

تعرف على وثائق XML

من الممكن ان تكون وثيقة XML في صورة ملف مُخزن على الكمبيوتر ، أو فيض بيانات عبر الشبكة data stream (في الحقيقة نظرياً يمكنك أن تقرأ هذه البيانات عبر الشبكة ، لكن في الواقع تكون هذه البيانات مضغوطة في صورة ثنائية Binary)، كما أنها من الممكن أن تكون سلسلة نصية تُعالج في ذاكرة الحاسب ، يجب أن تكون رصينة المحتوى متكاملة في المعنى حتى بدون الـ Schema فحتما ستتبع قواعد معينة في كتابتها ، حتى في بساطة الوثيقة المكتوبة أياً كانت ، يجب أن تتبع قاعدة عامة في غلق كافة الـ فروع (childes) قبل أي وسم (tag) رئيسي (parent) ، على سبيل المثال تعتبر الوثيقة التالية منسقة تماماً:

```
<states>
    <state>
        <name>Dakahli</name>
        <city>Mansoura</city>
        <city>Belqas</city>
    </state>
</states>
```

فكما ترى الجذر (root) يحوي العنصر states موضوع بين الوسم <states> وانتهى بالوسم </states> ، هذا هو مانسميه العنصر الرئيسي parent للعنصر الفرعي state الذي بدوره يعتبر parent للعنصر name والعنصرين city ،أي وثيقة XML تحوي عنصر root واحد فقط ، ربما تجد بعض العناصر تحتوي على خصائص attributes ، ففي المثال التالي تم وضع خصيصة name للعنصر state:

```
<states>
  <state name="Dakahli">
      <city>Mansoura</city>
      <city>Belqas</city>
  </state>
</states>
```

هذا المثال كما السابق يعود بنفس البيانات فقط أسندنا الخاصية name للعنصر state دون أن يكون لها وسم مستقل .
أي عنصر يُمكن ان يحتوي عدد لا محدود من الخصائص شريطة ألا تكون هذه الخصائص متكررة ، فمثلا العنصر city لن يتم وضعه كخصيصة هنا لأنه متكرر مرتين لمدينتين فالأحرى وضعه داخل وسم كما في المثالين أعلى ، كما أنه قد يحتوي على بيانات (نصوص أو حتى عناصر أخرى) أو قد يكون فارغاً يعني يبدأ ويغلق الوسم دون فروع (<......./>)، مثال على ذلك إذا أردت أن تعرف عدد المحافظات ، يمكنك استخدام عنصر فارغاً ليحل لك هذه المسألة:

```
<states>
   <controlinfo count="1"/>
    <state name="Dakahli">
       <city>Mansoura</city>
       <city>Belqas</city>
    </state>
</states>
```

كما تلاحظ العنصر الفارغ controlinfo يحوي خاصية واحدة فقط count لكن بلا فروع ، لأننا بدانا الوسم وأغلقناه في نفس سطر العنصر <controlinfo count="1"/> ويمكن أيضاً تمثيل هذا الوسم بهذه الطريقة:

```
<controlinfo count="1"> </controlinfo>
```

ملحوظة:
على الرغم من سهولة تصميم وثائق XML ، إلا أنه يحتاج بعض الجهد كما تصمم قواعد البيانات.
بعض مصممي XML لايتفقون علي أهمية الخصائص attributes اذ يمكنك الاستعاضة عنها بالوسوم كما رأينا (وبالتالي إن لم تظهر لها اهمية فلن يكون هناك أهمية للعنصر الفارغ ككل) ، ولكنهم يرون أنها تشبه في هذه الحالة قواعد البيانات العلائقية على كل حال لا يعني هذا أن الخائص لن تجد لها مكاناً في تصميم وثيقة XML فليس مطلوباً من XML أن تطابق كافة خصائص قواعد البيانات العلائقية.

فهم XML Declaration

في حالة أردت أن تتعمق فى تطوير تطبيقات XML فانت لن تتوقف على استخدام العناصر والخصائص وفقط ، فإذا أردت ان توافق معايير W3C عليك أن تُضمِّن XML Declaration في وثيقة الـXML ومكانها يقبع أعلى العنصر الجذري root element في وثيقة XML.
طريقة كتابة الـ declaration مثل كتابة العناصر لكن مايميزها أننا نضع علامة استفهام بجوار فتحة الوسم ، وتحوي الخاصية version; والتي تمثل بالقيم 1.0 , 1.1 ، ولتوضيح الطريقة التي تُكتب بها انظر التالي:

```
<? Xml version="1.0" ?>
```

هناك الكثير والكثير من الخصائص التي تمتلكها XML لكن ما ذكرناه آنفاً هو كاف لتبدأ معها ، فكما رأيت لم نستخدم أي تصريحات خاصة باللغة او schemas أو namespaces ، والمثال الذي طرحناه هو مثال متكامل يمكننا فيما بعد التعديل والاضافة عليه ليصبح مثالاً حقيقياً.

تحويل البيانات العلائقية إلى XML

استخدمنا الاستعلام SELECT لكي نعود بقيمة عمود ما في قاعدة بيانات ، يُمكننا أن نسترجع قيمة استعلام مكتوب بالـ SQL الي XML وذلك عن طريق الاستعلام FOR XML بإضافته الى الجملة SELECT ، وعن طريق MS SQL SERVER 2008 نستطيع زيادة قدرة هذا الاستعلام والحصول على نتائج لإستعلامات معقدة بزيادة قدرة الاستعلام عن طريق الكلمات المفتاحية. يُستخدم الاستعلام FOR XML لتحويل النتائج من الاستعلام العادي الى بنية XML ، وتوفر أربعة أنماط لذلك:

- FOR XML RAW
- FOR XML AUTO
- FOR XML PATH
- FOR XML EXPLICIT

سنستخدم اول نمطين في امثلة لتوضيح ذلك.

اولاً FOR XML RAW:

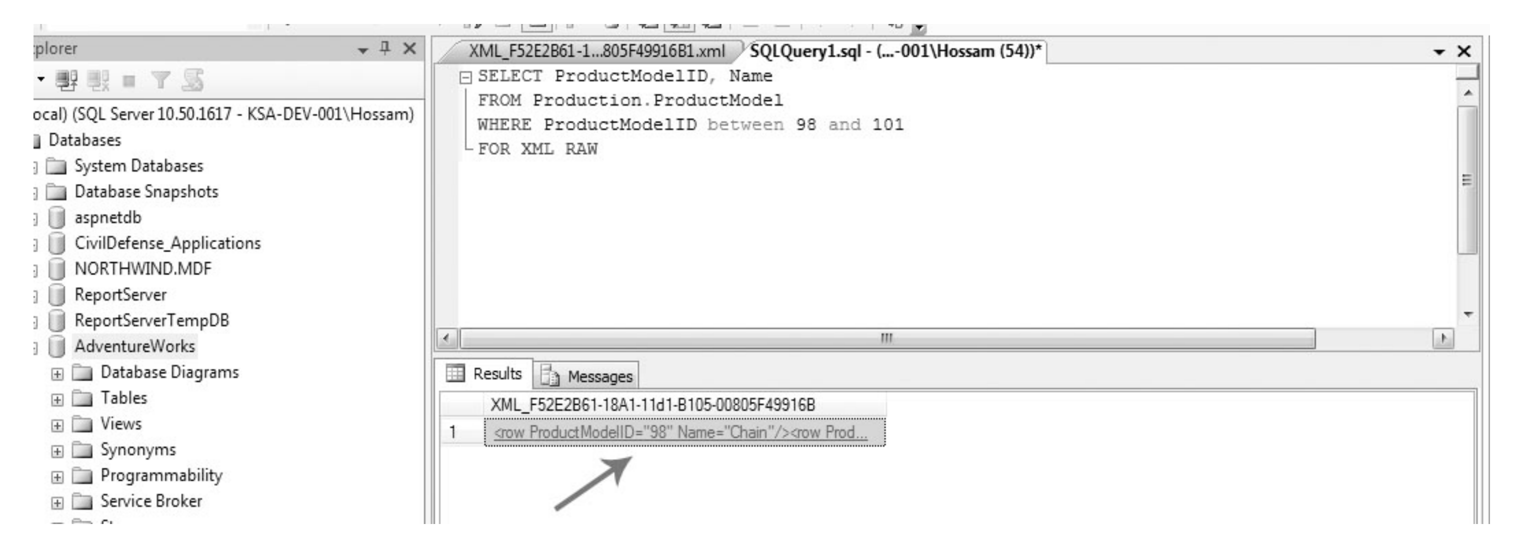
يستخدم هذا النمط لتحويل كل صف في الاستعلام العادي الى عنصر XML ، هذا العنصر يُعرف كصف في ناتج الاستعلام، وكل عمود يتم الاستعلام عنه بجملة SELECT يتم توصيفه كخاصية ATTRIBUTE داخل عنصر هذا الصف ، يحدث هذا مع الاعمدة التي لا تساوي NULL.

جرب هذا المثال:

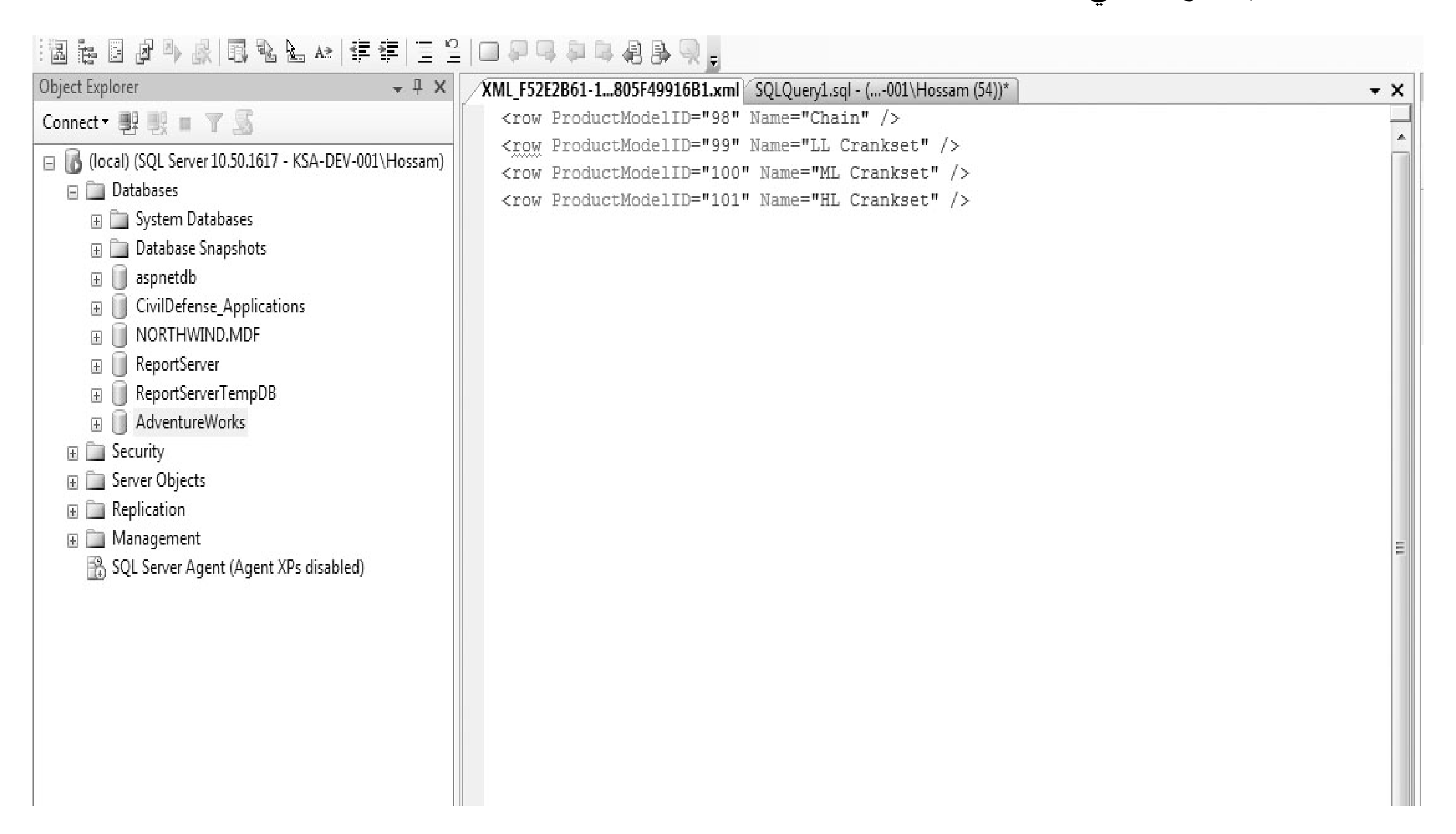
- افتح MS SQL Server كما تعودت.
- في مستعرض العناصر قم باستعراض قواعد البيانات عندك واختر قاعدة البيانات AdventureWorks ثم كلك يمين واختر New Query.
- أدخل الاستعلام التالي ، ثم اضغط تنفيذ:

```
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID between 98 and 101
FOR XML RAW
```

لترى النتيجة كما بالشكل ، عبارة عن رابط في نافذة النتائج ،



اذا ما ضغط عليه سترى التالي:



كما أسلفنا في شرح وظيفة هذا الاستعلام فإنه كما رأيت يقوم بتحويل صف الاستعلام المعتاد إلى صف في وسم الـ XML ،ثم قام بالاستعاضة عن القيم التي كانت ستظهر في الاستعلام العادي الى خصائص ، باستخدام الأسماء المستعارة Alias Names التي حددناها في الاستعلام ،لينتج مصفوفة من العناصر كما رأيت بالشكل ، لا يمكن بحال أن نسمي الاستعلام الناتج على أنه وثيقة XML فلو تذكر أنه لكي يكون عندنا وثيقة XML فإنه يلزمها عنصر واحد يعمل كجذر root ، وهو مالا يتوفر في الاستعلام السابق.

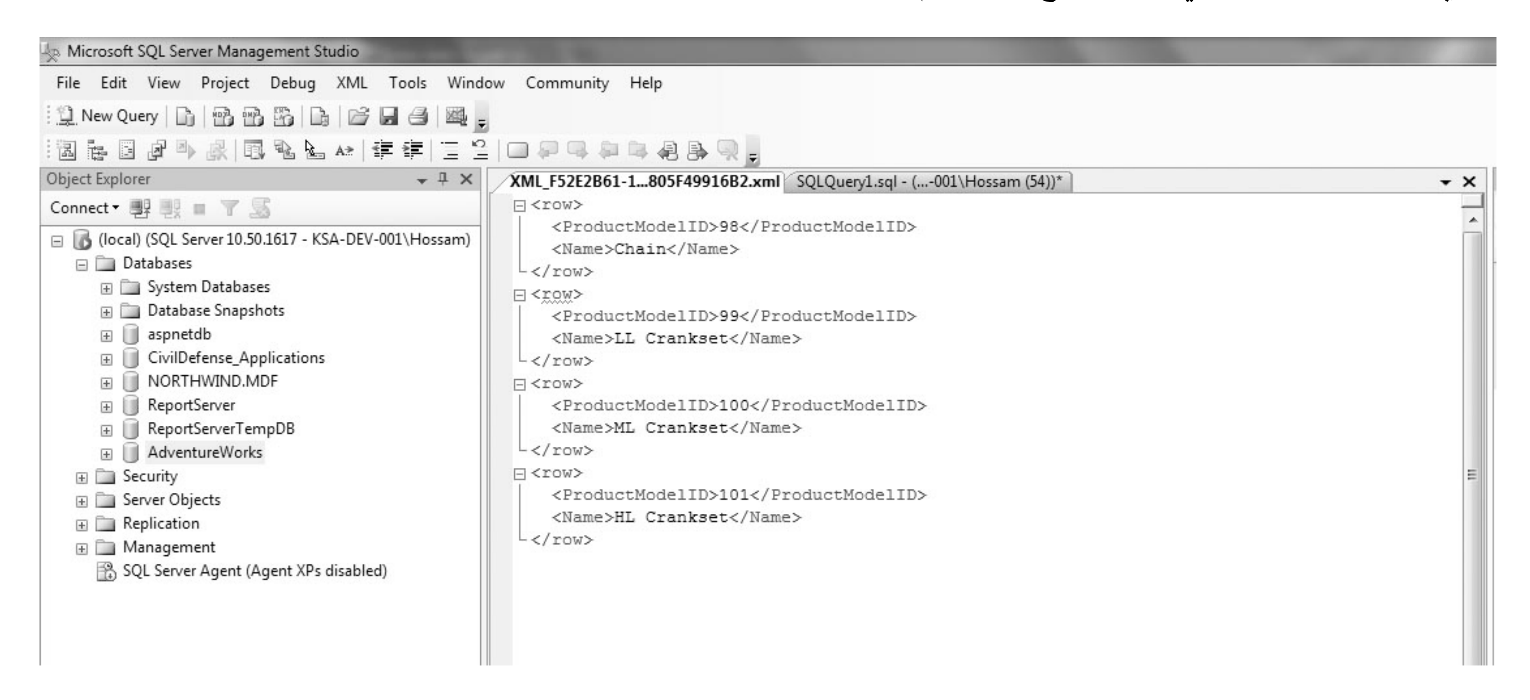
جرب المثال التالي أيضاً:

في المثال السابق ، كان التركيز على الخصائص attributes في المثال هذا سيتم التركيز على العنصر element ، مما يعني انه سيتم انشاء عناصر جديدة لكل عمود في الاستعلام ، ولعمل ذلك لابد من اضافة الكلمة المفتاحية ELEMENTS بعد الاستعلام FOR XML RAW كما في المثال:

- نفذ الخطوات السابقة لعمل استعلام جديد
- قم باضافة الاستعلام التالي ، ثم اضغط F5:

```
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID between 98 and 101
FOR XML RAW,ELEMENTS
```

لتري كما بالمثال السابق في نافذة النتائج رابط ، قم بالضغط عليه لترى كما بالشكل:



كما رأيت فوضعك للكلمة المفتاحية ELEMENTS بجوار الاستعلام FOR XML RAW هي كلمة السر في تبدل الناتج عن السابق ، لأنها ببساطة تحول قيمة كل عمود إلى عنصر مستقل بدلاً من خاصية كما بالسابق ، لهذا فإنها تستخدم كـ (element-centric) ، وهي أيضا كسالفتها لم تولد وثيقة XML لعدم احتواءها على root.

كيفية اعادة تسمية الصفوف

يُمكننا في الاستعلامات السابقة اعادة تسمية الصفوف الناتجة ، عن طريق وضع خصائص اختيارية للاستعلام السابق ، كما في المثال التالي ، سنضع اسماء مستعارة للصفوف في الاستعلام FOR XML RAW:

- قم بالكتابة فوق الاستعلام السابق بوضع الاستعلام التالي محله واضغط EXECUTE:

```
SELECT ProductModelID, Name
FROM Production.ProductModel
WHERE ProductModelID between 98 and 101
FOR XML RAW ('ProductModelDetail'),ELEMENTS
```

- لترى رابط في نافذة النتائج بالضغط عليه ترى نتيجة كما بالشكل التالي:

```
XML_F52E2B61-1...805F49916B3.xml   SQLQuery1.sql - (...-001\Hossam (54))*
<ProductModelDetail>
  <ProductModelID>98</ProductModelID>
  <Name>Chain</Name>
</ProductModelDetail>
<ProductModelDetail>
  <ProductModelID>99</ProductModelID>
  <Name>LL Crankset</Name>
</ProductModelDetail>
<ProductModelDetail>
  <ProductModelID>100</ProductModelID>
  <Name>ML Crankset</Name>
</ProductModelDetail>
<ProductModelDetail>
  <ProductModelID>101</ProductModelID>
  <Name>HL Crankset</Name>
</ProductModelDetail>
```

كما رأيت فإن الاستعلام FOR XML RAW ('alias') يقوم بالاستعاضة عن الاسم raw بأي اسم مستعار تضعه بين الاقواس ، اذا لم تضف الكلمة المفتاحية ELEMENTS للإستعلام السابق ، فسوف يتم تغيير الناتج ليصبح attribute centric مع تغيير الاسم raw أيضاً:

```
<ProductModelDetail ProductModelID="98" Name="Chain" />
<ProductModelDetail ProductModelID="99" Name="LL Crankset" />
<ProductModelDetail ProductModelID="100" Name="ML Crankset" />
<ProductModelDetail ProductModelID="101" Name="HL Crankset" />
```

ملاحظات على FOR XML RAW

كما لاحظت فإن الناتج دائما لايكون وثيقة XML لعدم وجود العنصر الجذري ، كما لاحظت أيضاً ان الناتج إما يكون element centric، أو attribute centric ولا تستطيع أن تحصل على الاثنين معاً ، كما لاحظت أيضاً أن كافة عناصر XML في نفس المستوى الشجري لا يوجد PARENTS أو CHILDES.

ثانياً FOR XML AUTO

تستخدم هذه الخاصية لإرجاع استعلام XML متشابك nested لهذا لن نجد المزيد من التحكم في الاستعلام الناتج ، وتظهر فائدتها جلياً اذا أردت أن تسترجع متسلسلات hierarchies بسيطة.

كل جدول يتم الاستعلام عنه بالجملة FROM وعلى الأقل يتم الاستعلام عن عمود بجملة SELECT يتم تمثيله كعنصر XML ، اما في صورة عنصر فرعي subelements، أو في صورة خاصية attribute .

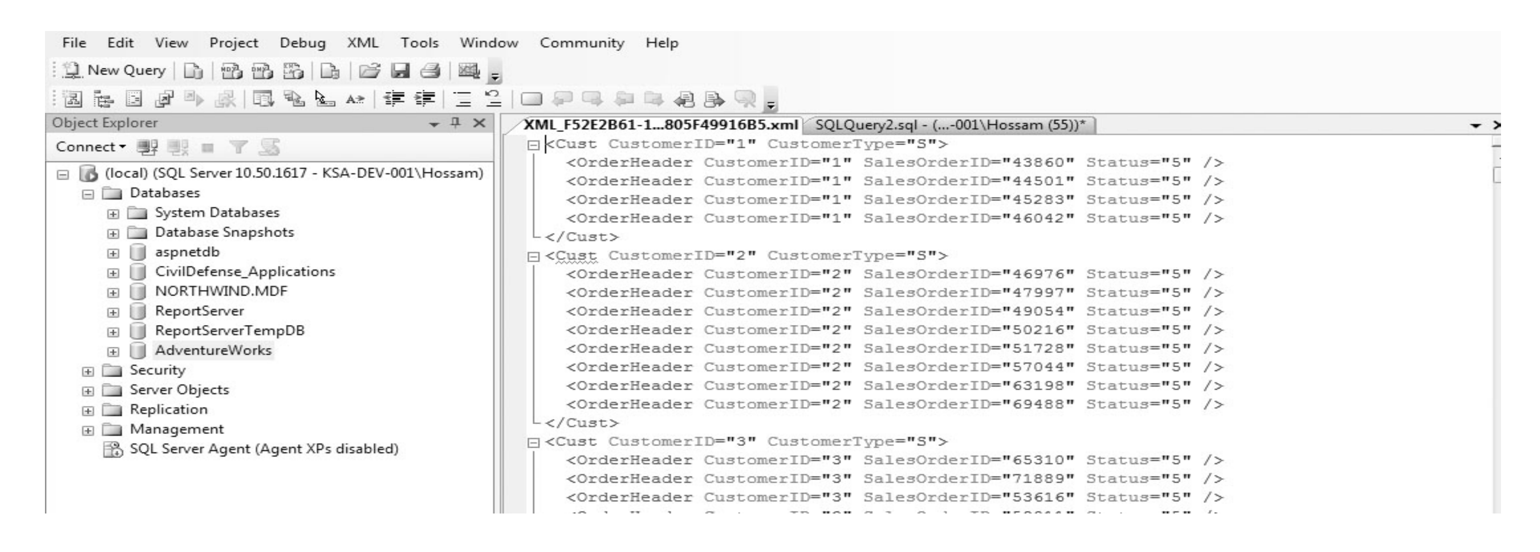
مثال على FOR XML AUTO

اتبع الخطوات التالية لتنفيذ الاستعلام الخاص بـ FOR XML AUTO:

- قم بفتح نافذة استعلام جديد في قاعد البيانات AdventureWorks وادخل الاستعلام التالي واضغط execute:

```
SELECT Cust.CustomerID,
OrderHeader.CustomerID,
OrderHeader.SalesOrderID,
OrderHeader.Status,
Cust.CustomerType
FROM Sales.Customer Cust, Sales.SalesOrderHeader
OrderHeader
WHERE Cust.CustomerID = OrderHeader.CustomerID
ORDER BY Cust.CustomerID
FOR XML AUTO
```

لترى في نافذة النتائج رابط قم بالضغط عليه لترى كما بالشكل:



كما لاحظت فإن الخاصية CustomerID تفهرس الجدول Cust ، لهذا تم انشاء عنصر باسم Cust وخاصية attribute باسم CustomerID ، أيضاً الثلاثة أعمدة OrderHeader.CustomerID , OrderHeader.SaleOrderID, OrderHeader.Status تفهرس الجدول OrderHeader ، لهذا تم اضافة العنصر OrderHeader كعنصر فرعي subelement للعنصر Cust وتم وضع الأعمدة الثلاثة كخصائص للعنصر OrderHeader.
تم أيضاً فهرسة العمود Cust.CustomerID للجدول Cust والذي كان معرفاً بنفس العمود ، لهذا لن يتم اضافة عناصر جديدة ،وبدلاً من ذلك تم اضافة الخاصية CustomerType للعنصر Cust الذي تم انشاؤه مسبقاً.
يقوم الاستعلام بتخصيص أسماء مستعارة للجداول ، تستعمل محل أسماء العناصر ، أما الـ الاستعلام ORDER BY فيستخدم لتجميع كل العناصر الفرعية Children تحت عنصر فرعي واحد Parent.

ملاحظات على FOR XML AUTO
لا ينتج عن استخدام FOR XML AUTO أى وثيقة XML لكونها لا ينتج عنها أي عناصر جذرية root elements.
من الملاحظ أيضاً ان الناتج إما يكون element centric، أو attribute centric ولا تستطيع أن تحصل على الاثنين معاً مثل سابقتها ، كما أنها لا توفر خاصية اعادة التسمية كما في FOR XML RAW ، ومع هذا فهي جيدة جدا في استخدام أسماء الجداول والأعمدة والأسماء المستعاة aliases حال توفرها.

كيفية تخزين واسترجاع XML documents باستخدام أنواع بيانات XML

يوفر لنا SQL Server 2008 نوع بيانات Data Type جديد هو (*xml*) والذي لم يصمم فقط ليتعامل مع وثائق XML (والتي يكفيها أن تتعامل مع نوع البيانات الحرفي *character strings* إذ أنه كاف ليحوي اي قيمة حرفية) ولكن صُمم هذا النوع ليتعامل مع عمليات وثائق الـ XML المختلفة.
في حقيقة الأمر عندما يتم ترجمة Parsing وثيقة XML لايعني أن العملية قاصرة على قراءة فقط ، ولكن هناك عملية تحديث لهذه الوثيقة أيضاً (يتم تحويل وثيقة XML إلى DOM tree) ..
ماهي الـ DOM ؟
هي اختصار لـ Document Object Model وهي منصة تُمكن البرامج والأكواد من تحديث والاتصال بأي وثيقة XML وليس هذا فحسب بل العديد من أنواع الوثائق الأخرى مثل HTML أو أي محتوى هيكلي structured documents عن طريق توفير الخصائص والكائنات والدوال المختلفة لذلك.
يُمكننا من اجراء أي تعديل على وثائق XML من تحديث وحذف وإضافة قيم وعناصر من و إلى المتسلسلة الشجرية hierarchy.
ما سنقوم بعمله هو اجراء تحديث باستخدام xml data type واما عن طبيعة نوع البيانات هذا ومزاياه فهو خارج اطار الكتاب هذا ، لأنه يحتاج حقيقة الى كتاب منفصل ، كل احتياجنا منه هنا كمبرمج قواعد بيانات هوما يلزمك معرفته لاسترجاع وتخزين وثائق XML.

تنوية
هناك طرق عديدة لإجراء العمليات المختلفة على وثائق XML باستخدام ADO.NET وكذلك SQLXML (وهي ترقية للإصدار SQL Server 2000 للتعامل مع XML) كل هذا سيكون جيد لو لم يكن لدينا هذه الميزة في SQL Server 2005 / 2008 ، بالطبع في هذا الكتاب نعطي نبذة وبامكانك الاستزادة اذا أحببت التخصص ، فمجرد امكانية كتابة XML من خلال T-SQL فإنها تفتح أمامك آفاق وإمكانيات متعددة لا حصر لها ولكن اجتهد في البحث وتوظيف ماتعلمته في ذلك.

جرب هذا المثال لإنشاء جدول لتخزين XML
لإنشاء جدول يحوي وثائق XML ، قم بتنفيذ الاستعلام التالي في نافذة استعلام جديدة:

```
create table xmltest
(
xid int not null primary key,
xdoc xml not null
```

)

لترى رسالة نجاح الاستعلام (Command(s) completed successfully.).

هذا الاستعلام يُشبه الطريقة التي ننشئ بها الجداول في العادة ، وعلى الرغم من أن نوع البيانات xml يختلف عن باقي انواع بيانات SQL Server ، إلا أن طريقة انشاء الجدول وتعريف الأعمدة هي هي مثلما في SQL ، فيما عدا أن xml data type لايُستخدم كمفتاح رئيسي Primary Key.

مثال على اضافة وثائق XML إلى الجدول xmltest

- قم بإضافة الاستعلامين التاليين ، في نافذة الاستعلامات السابقة ، ثم اضغط execute:

```
insert into xmltest
values(
1,
'
<states>
<state>
<abbr>CA</abbr>
<name>California</name>
<city>Berkeley</city>
<city>Los Angeles</city>
<city>Wilmington</city>
</state>
<state>
<abbr>DE</abbr>
<name>Delaware</name>
<city>Newark</city>
<city>Wilmington</city>
</state>
</states>
'
)

insert into xmltest
values(
2,
'
<states>
<state abbr="CA" name="California">
<city name="Berkeley"/>
<city name="Los Angeles"/>
<city name="Wilmington"/>
</state>
<state abbr="DE" name="Delaware">
<city name="Newark"/>
<city name="Wilmington"/>
</state>
</states>
'
)
```

لتظهر لك رسالة نجاح الاستعلام في نافذة النتائج ((row(s) affected)1) مرتين ، واذا أردت أن تستعرض نتيجة ماقمنا به ، فاكتب الاستعلام التالي في نافذة الاستعلامات واضغط F5:

Select * from xmltest

لترى في نافذة النتائج عمودين يحوي كل منهما وثيقة XML ، اضغط على العمود الأول في xdoc لترى كما بالشكل:



كما رأيت في السابق حين تعاملت مع الجملة INSERT لا يوجد أي اختلاف هنا ، قمنا بسناد قيمة المفتاح الرئيسي في عملية الادخال P.K ، ووثيقة XML كمتسلسلة نصية ونتج ما توقعناه تماماً.

خاتمة الفصل

في هذا الفصل قمنا بتغطية أساسيات XML والتي يستوجب على كل مبرمج C# أن يعرفها ، وتعرفنا على خصائص وثائق XML وكيفية التعديل فيها ، وكذلك كيفية انشاءها من خلال SQL Server، وكما نوهت الطريق أمامك الآن مفتوحاً للتعمق أكثر وأكثر في هذه التقنية وكيفية توظيف ذلك في برامجك المختلفة ومنصاتك المتعددة من الحواسب العادية الى أجهزة الموبايل ، كما تشاء.

الفصل السادس

العمليات Transactions

تلعب العمليات دوراً رئيسياً اليوم في عالم الأعمال لاشتمالها على العديد من الإجراءات والعمليات الأخرى ، وتظهر فوائدها جليا في تحقيق تكاملية البيانات Data Integrity للعديد من العمليات الأخرى المرتبطة ، وعندما يحدث تداخل فيما بين المستخدمين أثناء تعاملهم مع البيانات.
في هذا الفصل سنتحدث عن مبادئ الـ Transactions وكيفية استخدامها مع SQL Server 2008 ولغة C# ، وهذا هو المحتوى:

- ماهي الـ Transactions ؟
- متى نستعمل الـ Transactions؟
- فهم الخصائص ACID
- تصميم الـ Transactions
- حالات الـ Transaction
- تعيين حدود الـ Transactions
- ماهي جمل T-SQL المسموح بها في الـ Transaction؟
- العمليات المحلية (Local Transactions) في SQL
- العمليات الموزعة (Distributed Transactions) في SQL
- دليلك لكتابة أكواد عمليات فعالة
- كيفية برمجة العمليات
- كتابة أكواد للعمليات باستخدام ADO.NET

ماهي الـ Transactions ؟

هي مجموعة من الإجراءات والعمليات تُنفذ كحزمة واحدة ، فإما تنجح بالكلية أو تفشل بالكلية. ويُعد أشهر مثال لها هو عملية تحويل الأموال من الحساب الجاري إلى حساب التوفير في البنوك ، فهذه العملية تطلب إجرائين:
الأول عملية خصم الرصيد من الحساب الجاري ، والثاني إضافة المبلغ إلي حساب التوفير ، لذا لابد من نجاح العمليتين مرة واحدة وإلا فسوف يحدث عواقب وخيمة يترتب عليها خسارة الناس لأموالهم ، أو في حال الفشل لأي طارئ لاقدر الله يتم تدارك الأمر بايقاف كلا العمليتين وكان شيئاً لم يحدث عن طريق استخدام الـ Transactions ، فمنتهى العملية إما ضمان نجاح الاثنتين معاً وهو المرجو ، وإما فشل الاثنتين معاً وهو أخف الضررين اذ يبقى حسابك كما هو دون تغيير.

متى نستعمل الـ Transactions ؟

كما تبين من مثالنا فأفضل استعمال لها حينما نريد ضمان نجاح او فشل أكثر من عملية كوحدة واحدة ، الأمثلة التالية توضح بعض سيناريوهات استخدام الـ Transactions:

- في حالة معالجة الدفعات batch processing حينما نريد إدخال أو تعديل أو حذف أكثر من صف دفعة واحدة.
- في حالة أي تغيير على جدول يتطلب بقاء الجداول الآخرى كما هي.
- حينما نريد التعديل على أكثر من قاعدة بيانات بشكل متزامن.
- في حالة العمليات الموزعة على أكثر من قاعدة بيانات على أكثر من خادم Distributed Transactions.

وقت إجراء هذه العملية ، فإن قاعدة البيانات يضاف اليها قفل lock ، مما يجعل اجراء أي عملية أخرى على قاعدة البيانات غير ممكن ، الى أن يتم رفع هذا القفل ، عملية القفل هذه تبدأ من قفل للصف إلى قاعدة البيانات كلها. تُسى هذه العملية بالتزامن *concurrency* وتعني إمكانية التعامل مع أكثر من عملية على قاعدة بيانات في نفس الوقت.
في مثال البنك السابق ، القفل يضمن أن عمليتين منفصلتين لا تتمان على نفس الحساب في نفس الوقت ، فإما سحب او ايداع ، وإلا فكلاهما قد يُفقد.

ملحوظة:
استخدامك للعديد من الأقفال قد يؤثر بالسلب على أداء قاعدة البيانات ، لهذا يُفضل تعليق قاعدة البيانات من أجل الـ transactions يتم في مدي قليل من الزمن تجنباً لهذه المشكلة.

فهم الخصائص ACID

هناك أربعة خصائص للـ transactions تُختزل فى الكلمة ACID وهي :
الذرية Atomicity
الثباتية Consistency
العزل Isolation
البقائية Durability
ماذا يعني كل هذا ؟

- يُعتبر الـ transaction في الخاصية Atomicity حدثاً مستقلاً single action وليس دفعة من العمليات المنفصلة ، لذا حين تنجح كل هذه العمليات المستقلة ، فإن الـ transaction يُعتبر قد نجح ، وأما لو فشلت عملية واحدة من هذه العمليات ، فيُعتبر كل شئ كأن لم يكن ، ويحدث عملية roll back ، في مثال الـ order-entery في قاعدة البيانات Northwind ، حينما تقوم بإدخال order للجدولين orders – order details فإما ان يتم حفظهما معاً او حال الفشل يفشلا معاً.
- أما الخاصية Consistency فهي تعني الابقاء على قاعدة البيانات في حالة ثابتة سواءاً نجح الـ transaction أم لا ، ولكي تحقق تكاملية البيانات data integrity فلابد أت يتسق الـ transaction مع أي قيود مفروضة على أي أعمدة في الجدول ، في مثال قاعدة البيانات Northwind لايمكنك عمل أي شئ على الصفوف في الجدول Order Details من دون إجراءها على الصفوف المقابلة في الجدول Orders ، لأن هذا يجعل البيانات في حالة غير ثابتة inconsistent.
- الخاصية Isolation وتبين أن لكل transaction حدود مُعَرفة ، بما يعني عزل كل عملية عن الأخرى فلايتأثر عمل عملية بعمل العملية الأخرى ، فلو هناك عملية تقرأ بيانات الآن ، لايُمكن بحال أن تتحصل عملية أخرى عن طبيعة هذه البيانات إلا بعد الانتهاء من العملية الأخرى أو قبلها أما في اثناءها فلا.

- الخاصية Durability: أي تعديل ناجم عن نجاح transaction يتم حفظه بشكل دائم في النظام بغض النظر عن حدوث أي شئ آخر، كما يتم تسجيل (logging) هذه العمليات ، لذا فأي خطأ ينجم عنها يتم تداركه إلى حالته الأصلية قبل هذا الخطأ ، فبنهاية الـ transaction يتم تسجيل الصف في transaction log لقاعدة البيانات ، وفي حالة حدوث الخطأ ما في نظام قاعدة البيانات يتم استعادة قاعدة البيانات من backup ومن ثم يُمكننا استعادة الـ transaction المتحقق مسبقاً عن طريق هذا الـ transaction log.

ملحوظة:

يتم فرض هذه الخصائص بصفة آلية على أي خادم قواعد بيانات .

تصميم الـ Transactions

أغلب العمل على الـ transactions مستوحى من أمثلة حقيقية مثل المعاملات البنكية ، حجوزات الطيران حولات مالية ...إلخ ، لهذا كان الغرض من عملية تصميم (design) الـ transaction هو ابراز الخصائص الأساسية لهذه العمليات لتنقل هذه العمليات من الأنظمة اليدوية إلى الأنظمة الالكترونية بوضوح ، وتتضمن هذه العملية مايلي:

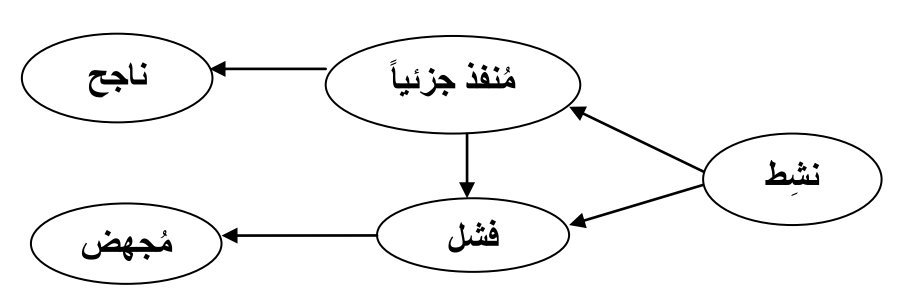
- ماهية قاعدة البيانات المستخدمة لهذا الـ transaction
- الخصائص الوظيفية للـ transaction
- ناتج الـ transaction
- دور المستخدمين
- معدل الإستخدام المتوقع

وهناك ثلاثة أنواع من الـ transactions:

١- Retrieval transaction وهو معني باسترجاع بيانات من الشاشة المعروضة
٢- Update transaction يقوم بإدخال سجلات جديدة او حذف قديمة أو التعديل في سجلات موجودة في قاعدة البيانات
٣- Mixed transaction وهو خليط مابين النوعين السابقين

حالات الـ Transaction

في حالة عدم حدوث أي مشكلة ، يتم تنفيذ الـ transaction بنجاح ، وكما أسلفنا لابد من تمامية عمل الـ transaction حتى يقال عنه ناجح ، يعتبر الـ transaction الغير مكتمل مجهضاً aborted اذا لم يقم بالوظيفة بشكل تام حتى ولو قبل النهاية بقليل ، ويُعتبر ناجحا committed إذا تم تنفيذه بالكلية ، الشكل التالي يوضح هذه الحالات:



تعيين حدود(Boundaries) الـ Transactions

تُساعدنا حدود الـ transaction على معرفة متى تبدأ وتنتهي الـ SQL Server transactions بمعرفة دوال وخواص الـ API وكذا جمل T-SQL كما في التالي:

- جمل T-SQL: لتوصيف الـ transaction استخدم الجمل التالية :

BEGIN TRANSACTION, COMMIT TRANSACTION, COMMIT WORK, ROLLBACK TRANSACTION, ROLLBACK WORK, SET IMPLICIT_TRANSACTIONS

وتستخدم هذه الجمل مع T-SQL scripts كمثل تلك التي تتنفذ باستخدام osql التي تنفذ باستخدام محث الأوامر command prompt utitlity .

- دوال وخواص API : حيث تحتوي على DB APIs مثل ODBC , OLE DB, ADO كما تحتوي على NET. SQLClient namespace يحوى هو الآخر دوال ووظائف تُستخدم لتوصيف الـ transaction.

يتم إدارة الـ transaction عن طريق أي من هذه الخواص بشرط أن تكون خاصية واحدة فقط وإلا سيحدث نتائج غير مرغوبة، على سبيل المثال لايُمكنك استخدام وظائف ODBC API لكي يبدأ الـ transaction في حين أنك تستخدم T-SQL COMMIT لكي تنهيه ، فهذا لن يُخطر مُشغل SQL Server ODBC أن الـ transaction قد تم ، لذا عليك أن تستخد ODBC SQLEndTran لكي تنهي الـ transaction.

ماهي جمل T-SQL المسموح بها في الـ Transaction؟

مسموح لكل استعمال كافة جمل T-SQL المختلفة في التعامل مع الـ transactions فيما عدا الجمل التالية:
ALTER DATABASE, RECONFIGURE, BACKUP, RESTORE, CREATE DATABASE, UPDATE STATISTICS ،DROP DATABASE

كما لايمكنك استخدام الإجراء sp_dboption لتغيير أي خيارات في قاعدة البيانات أو استخدام أي إجراءات مخزنة في تعديل قاعدة البيانات الرئيسية master في الـ tranasctions الصريحة أو الضمنية.

العمليات المحلية (Local Transactions) في SQL

تدعم كافة محركات قواعد البيانات بنية ذاتية خاصة لدعم الـ transactions ، وتسمى الـ transactions التي يتم قصرها على قاعدة بيانات واحدة أو مصدر واحد بالـ transactions المحلية ، ويمكن ان توجد في أربعة أنماط كما بالتالي:

- Autocommit Transactions يعتبر هذا النمط هو الرئيسي لإدارة الـ transaction داخل SQL Server فيتم حفظ كل جملة T-SQL عند الإكتمال بنجاح أو التراجع عنها في حالة حدوث أي اخطاء ، ويظل هذا النمط افتراضياً مالم يتم التعديل على اي من أنواع الـ transactions.
- Explicit Transactions وهو النمط الذي تتحكم فيه بشكل مباشر عند بداية أو نهاية الـ transaction وكان هذا النمط يُسمى في الإصدار ٢٠٠٠ من SQL Server باسم *user-defined transaction* ، ويتم استخدام جمل T-SQL التالية في هذا النمط:BEGIN TRANSACTION, COMMIT TRANSACTION, ROLLBACK TRANSACTION ، ويعمل هذا النمط حال مدة بقاء الـ transaction فعند الإنتهاء يعود الاتصال عند آخر نمط للـ transaction قبل استخدام هذا النمط المباشر.
- Implicit Transactions يتم حفظ أي استعلام تم تنفيذه في صورة DML تقوم به حينما تتصل بقاعدة بيانات باستخدام MS SQL Server Management Studio بطريقة آلية ، ويحدث هذا افتراضيا مادام الاتصال في نمط AutoCommit فإذا ما أردت أن تجعل هذه التغييرات ضمنية ، فما عليك إلا أن تضع الاتصال بقاعدة البيانات في الوضع Implicit Transaction عن طريق الاستعلام:

```
SET
IMPLICIT _TRANSACTIONS ON|OFF
```

وبعد وضعها في هذا النمط يقوم SQL Server ببدء الـ transaction بطريقة آلية عندما يتم تنفيذ أياً من هذه الجمل:
ALTER TABLE, CREATE, DELETE, DROP, FETCH, GRANT, INSERT, OPEN, REVOKE, SELECT, TRUNCATE TABLE, UPDATE.

ويظل الـ transaction فعالاً حتى يتم التصريح بالجملة COMMIT أو ROLLBACK ،فمثلاً عندما يتم تحديث سجل ما في قاعدة البيانات باستخدام الجملة UPDATE فيقوم SQL Server بقفل قاعدة البيانات تلك الى أن يتم التصريح بأي من الجملتين السابقتين ويتم التراجع آليا عندما يقوم المستخدم بقطع الاتصال بقاعدة البيانات في حالة عدم استخدامك أياً من الجملتين السابقتين ، ولهذا لايُفضل استخدام هذا النمط مع قواعد البيانات ذات التزامنية العالية.

- Batch-Scoped Transactions في هذا النمط يعمل الـ transaction على اخراج مجموعة من النتائج المتعددة فيما يعرف بـ Multiple Active Result Sets واختصاراً بـ MARS ، فهذه الخاصية تُمكن الـ transaction من تنفيذ أكثر من أمر نشط لأنها تحوي بيئة جيدة لدعم تنفيذ العمليات المتتابعة عن طريق كائنات ADO.NET ، فما يحدث هو أن أي تغيير يتم تركيزه على حزمة batch حتى تنتهي تماما ، ويتم نسخ اعدادات التنفيذ الى البيئة الافتراضية ، لذا هذا النمط يتم فقط حينما يتم تمكين MARS وهناك أكثر من حزمة يتم تنفيذها معاً ، وحتى لايحدث التباس فإن MARS تنفذ أكثر من حزمة وليس أكثر من transaction.

العمليات الموزعة (Distributed Transactions) في SQL

على نقيض العمليات المحلية local transaction التي تقتصر على مورد واحد او قاعدة بيانات واحدة ، تشتمل العمليات الموزعة Distributed Transactions على سرفرين أو أكثر ويطلق عليهم *resource managers* ويتم التناغم بينهم عن طريق مكون خدمي يُسمى *transaction manager* أو *transaction coordinator* ، يستطيع SQL Server أن يعمل كمدير للموارد للعمليات الموزعة مدعوماً بـ transaction managers مثل Microsoft Distributed Transaction Coordenator (MS DTC).
الـ transaction ذو خادم قواعد البيانات الواحد والذي يعمل على قاعدتي بيانات أو أكثر يُعامل كـ distributed transaction وتتم إدارة هذه العملية الموزعة داخليا في SQL Server.
يتم إدارة العمليات الموزعة على مستوى التطبيقات بنفس الآلية التي يدار بها العمليات المحلية ، ففي نهاية تنفيذ الـ transaction يتطلب منه تحديد الحالة مابين التمام أو التراجع ، في حالة التمام في العملية الموزعة يتم إدارتها بطريقة مختلفة حتى يتم تقليص الخطر الناجم عن أي مشاكل شبكية وهذه الطريقة تُسمى *two phase commit* وهى:
مرحلة التمهيد Prepare Phase: عندما يتسلم مدير العمليات طلب بالتمام ، يقوم بارسال أمر تمهيد الى كافة مديري الموارد المستخدمة في العملية ، وكل مدير مورد بدوره يعمل المطلوب التى تجعل العملية durable ، وأي بيانات مؤقتة استخدمت في تسجيل خطوات هذه العمليات يتم حفظها فى القرص ، وبنهاية هذه المرحلة يتم الرجوع بالنتيجة ناجحة او العملية فشلت لمدير العملية.
مرحلة التسليم Commit Phase: اذا تسلم مدير العملية رسالة نجاح عملية التمهيد من كافة مديري الموارد ، يتم ارسال أمر تسليم الى كل مدير مورد ، في حالة ارسال رسالة نجاح من كل مديري الموارد ، يقوم مدير العملية بإرسال تنوية على ناجح العملية للتطبيق ، أما لو تم ارسال رسالة اخفاق من أي مدير للموارد فيتم على تراجع للعملية برمتها ويتم ارسال حالة فشل للتطبيق.

دليلك لكتابة أكواد عمليات فعالة

هذه بعض الاقتراحات من اجل كتابة أكواد للعمليات كي تعمل بكفاءة:

- لا تُجبر المستخدم على مدخلات أثناء العملية ، اذا أردت ذلك عليك بملئ هذه المدخلات قبل بدء العملية اما لو تُطلب ذلك أثناء العملية فما عليك إلا عمل تراجع عن العملية roll back ثم دع المستخد يضيف المدخلات اللازمة ، ثم أعد تنفيذ العملية مرة اخرى ، والسبب في ذلك أن هذه العمليات تتم في كسور من الثانية مما يعوقها بطئ المستخد في مدخلاته فينجم عنها اخطاء غير مرغوبة.
- لا تقم بفتح عملية أثناء معالجتها لبيانات ، فالمفترض ان أية عملية لاتبدأ قبل نهاية عملية التحليل التحضيري للبيانات.
- حافظ على كون العملية قصيرة بقدر المستطاع ، اجعلها فقط ببداية العملية ثم تنفيذ تعديل ، ثم اصدار رسالة نجاح او اخفاق للعملية.
- حاول أن تستخدم أقل قدر من البيانات أثناء العملية ، لأن هذا بدوره سيقلل عد الصفوف المقفولة مما يقلل من التضارب فيما بين العمليات.

كيفية برمجة العمليات

نستخدم الجمل الثلاثة التالية لكي نتحكم في الـ transactions في SQL Server:

- BEGIN TRANSACTION: وهي لتحديد بداية العملية.
- COMMIT TRANSACTION: وهي لتحديد نجاح نهاية العملية ، وتعطي اشارة لقاعدة البيانات لحفظ العمل.
- ROLLBACK TRANSACTION: وهي لتوضيح أن العملية أخفقت كما وتعطي اشارة لقاعدة البيانات للتراجع الى الحالة التي تسبق هذه العملية.

لاحظ انه لا يوجد جملة END TRANSACTION لأن العملية تنتهي ضمنياً أو تصريحياً في حالة النجاح و الإخفاق.

برمجة العمليات باستخدام T-SQL

سنقوم باستخدام الإجراءات المخزنة لنتدرب على برمجة العمليات باستخدام مثال غير حقيقي لتتعرف على الأساسيات وسنركز على طريقة عمل الـ transaction وما ستحتاجه بالفعل عندما تبرمجه باستخدام لغة الـ C#.

كما وننوه على أن استخدامك لجمل ROLLBACK , COMMIT داخل الإجراء المُخزن يتطلب الحرص من كون العملية بالفعل قيد التشغيل من جراء استدعاءها من هذا الاجراء المُخزن ، لا تشغل بالاً بطريقة عمل المثال ولكن ينبغي الحرص.

سنقوم الآن بكتابة transaction لكي يضيف ويحذف عميل من قاعدة البيانات Northwind في الجدول Customers ، هذا الجدول يحوي ١٢ عموداً ، منهم عمودان CustomerID و CompanyName لايسمحان بالقيم الفارغة null في حين الباقي يسمح بذلك ، لذا سنستخدم كلا العمودين في ادخال البيانات ، وسنستخدم العمود CustomerID في عملية ايجاد الصفوف عندما نستعرض الزبائن مرتبين حسب الـ ID.

- قم بفتح SQL Server Management studio كما تعودت وقم بالاتصال بخادم قواعد البيانات.
- قم بتوسعة قواعد البيانات في التصفح الشجري حتى ترى قاعدة البيانات Northwind إن لم تكن ظاهرة بالفعل ثم قم بفتح نافذة استعلام جديدة New Query.
- قم بانشاء اجراء مخزن stored procedure مستخدما الكود بالأسفل ثم اضغط F5 للتنفيذ:

```
create procedure sp_Trans_Test
@newcustid nchar(5),
@newcompname nvarchar(40),
@oldcustid nchar(5)
as
declare @inserr int
declare @delerr int
declare @maxerr int
set @maxerr = 0
begin transaction
-- Add a customer
insert into customers (101ustomerID, companyname)
values(@newcustid, @newcompname)
-- Save error number returned from Insert statement
set @inserr = @@error
if @inserr > @maxerr
set @maxerr = @inserr
-- Delete a customer
delete from customers
where 101ustomerID = @oldcustid
-- Save error number returned from Delete statement
set @delerr = @@error
if @delerr > @maxerr
set @maxerr = @delerr
-- If an error occurred, roll back
if @maxerr <> 0
begin
rollback
print 'Transaction rolled back'
end
else
begin
commit
print 'Transaction committed'
end
print 'INSERT error number:' + cast(@inserr as nvarchar(8))
print 'DELETE error number:' + cast(@delerr as nvarchar(8))
return @maxerr
```

- لكي تقوم بتجربة الاجراء الذي أنشاته قم بكتابة الاستعلام التالي أسفل الاستعلام السابق ، وقم بتظليله ثم اضغط Execute لتشغيل الاجراء:

exec sp_Trans_Test 'a ', 'a ', 'z '

لترى في نافذة النتائج رجوع النتائج بصفر كما بالشكل التالي:

```
rollback
print 'Transaction rolled back'
end
else
begin
commit
print 'Transaction committed'
end
print 'INSERT error number:' + cast(@inserr as nvarchar(8))
print 'DELETE error number:' + cast(@delerr as nvarchar(8))
return @maxerr

exec sp_Trans_Test 'a ', 'a ', 'z '
```

```
(1 row(s) affected)

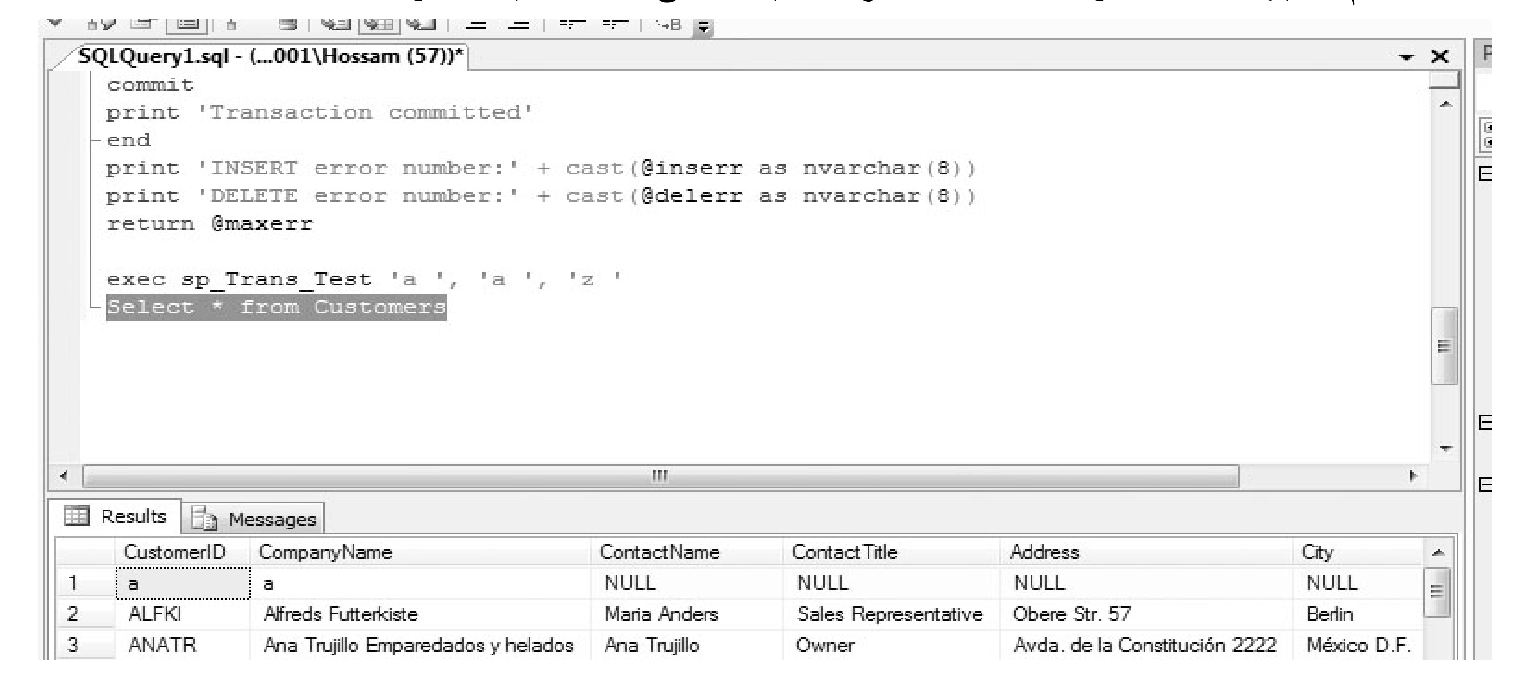
(0 row(s) affected)
Transaction committed
INSERT error number:0
DELETE error number:0
```

Query executed successfully. | (local) (10.50 RTM) | KSA-DEV-001\Hossam (57) | NORTHWIND.MDF | 00:00:00 | 0 rows

- في نافذة الاستعلام قم بإضافة الجملة التالية:

Select * from Customers

قم بتظليلها كما بالشكل واضغط Execute لترى العميل المسمى "a" قد أُضيف للجدول :



- قم بإضافة customer مستخدما القيمة "aa" لكلٍ من @newcustid و @newcompname والقيمة "z" للمعامل @oldcsutid عن طريق الجملة :

```
exec sp_Trans_Test 'aa ', 'aa ', 'z '
```

ثم اضغط تنفيذ لتحصل على رسالة كما بالشكل قبل السابق.

- حاول ان تستخدم جملة select كما بالسابق لترى أنه قد أضيف الزبون باسم "aa" فقط قم بتظليل جملة select السابقة وتنفيذها لترى النتيجة.

لشرح ماحدث فإننا قمنا بتعريف ثلاثة معاملات إدخال في الإجراء المُخزن :

```
create procedure sp_Trans_Test
@newcustid nchar(5),
@newcompname nvarchar(40),
@oldcustid nchar(5)
as
```

كما قمنا بتعريف ثلاثة متغيرات:

```
declare @inserr int
declare @delerr int
declare @maxerr int
```

لكي نستخدمها مع الإجراء المُخزن حيث تعود بعدد الأخطاء الحادثة جراء استخدام الجمل INSERT و DELETE.
قمنا بتحديد بداية العملية عن طريق الجملة BEGIN TRANSACTION واتبعناها بالجملتين INSERT و DELETE وبعد كل جملة يتم تخزين الرقم العائد :

```
begin transaction
-- Add a customer
insert into customers (customerid, companyname)
values(@newcustid, @newconame)
-- Save error number returned from Insert statement
set @inserr = @@error
if @inserr > @maxerr
set @maxerr = @inserr
-- Delete a customer
delete from customers
where customerid = @oldcustid
-- Save error number returned from Delete statement
set @delerr = @@error
if @delerr > @maxerr
set @maxerr = @delerr
```

انه لمن المهم استدراك الأخطاء في SQL Server خاصة في أكواد العمليات ، فلربما أثناء تنفيذك لأي جملة يحدث إخفاق ، وعن طريق الدالة @@ERROR يتم رجوع رقم الخطأ لآخر جملة تم تنفيذها ، وفي حالة خلو الجملة من الأخطاء تعود بالقمية "صفر".
بعد تنفيذ كل جملة T-SQL بنجاح يتم اعادة ضبط الدالة @@ERROR ، لذا اذا اردت أن تعيد قيمة الخطأ داخليا كما في حالتنا تلك ، فقط قم بتخزين رقم الخطأ في معامل ، قبل تنفيذ الجملة التي تليه لكي يُمكنك قراءتها فيما بعد.
في حالة رجوع الدالة @@ERROR بأية قيمة بخلاف الصفر ، فهذا يعني ان ثمة خطأ ما قد حدث ، وتحتاج الى التراجع عن العملية مرة أخرى ، قمنا أيضا بادراج الجملة PRINT لكي توضح ان العملية تمت بنجاح او تم التراجع عنها :

```
-- If an error occurred, roll back
if @maxerr <> 0
begin
rollback
print 'Transaction rolled back'
```

```
end
else
begin
commit
print 'Transaction committed'
end
```

تستخدم T-SQL العديد من الكلمات المحفوظة بتنويعاتها معاً ، في المثال السابق استخدمنا فقط الجملتين ROLLBACK و COMMIT.

قمنا باستخدام بعض الحيل ، لنتمكن من رؤية أرقام الأخطاء الحادثة أثناء العملية.

```
print 'INSERT error number:' + cast(@inserr as nvarchar(8))
print 'DELETE error number:' + cast(@delerr as nvarchar(8))
return @maxerr
```

لنرى ماحدث عندما قمنا بتنفيذ الاجراء المخزن:

قمنا بتشغيله مرتين ، المرة الأولي لإضافة الزبون "a" والمرة الثانية لإضافة الزبون "aa" ، وقمنا باضافتهم على الرغم من عدم وجودهم الى الجملة delete ، ولما يحدث اخفاق أو نجاح يحدث هذا بشكل موحد كوحدة واحدة ، اذاً لماذا حينما يحدث نجاح لعملية INSERT يتوقف عمل الجملة DELETE ؟

في نافذة النتائج التي رأينها في الشكل قبل السابق ، لعلك لاحظت ان قيمة error result = 0 ، والسبب في عدم الحذف على الرغم من عودة قيمة delete بالنجاح ، أن الصف غير موجود أصلاً ليتم حذفه ، وأما في حالة الاخفاق فإنه سيرى الصف السابق ليقوم بحذفه ، في الحقيقة لا يعامل هذا كخطأ في T-SQL، أما باقي الـ customersلايمكن حذفهم باستثناء مثاليا لأنهم يحوون فروع منهم child orders ، ذلك لايمكن حذفهم قبل حذف الفروع .

مثال لتجربة ماذا يحدث عن اخفاق العملية الأولى:-

في هذا المثال حاول ان تضيف زبون مضاف بالفعل ، ثم تحذف الآخر ، قم بإضافة الزبون "a" وقم بحذف الزبون "aa" عن طريق تنفيذ الاجراء التالي:

```
exec sp_Trans_Test 'a', 'a', 'aa'
```

لترى النتيجة كما بالشكل:

```
exec sp_Trans_Test 'a ', 'a ', 'z '
Select * from Customers
exec sp_Trans_Test 'aa ', 'aa ', 'z '
exec sp_Trans_Test 'a', 'a', 'aa'
```

Messages

```
Msg 2627, Level 14, State 1, Procedure sp_Trans_Test, Line 12
Violation of PRIMARY KEY constraint 'PK_Customers'. Cannot insert duplicate key in object 'dbo.Customers
The statement has been terminated.

(1 row(s) affected)
Transaction rolled back
INSERT error number:2627
DELETE error number:0
```

هل لاحظت تغير رسالة النتائج الى رقم الخطأ 2627؟

واما أمر الحذف فعاد برقم الخطأ 0 ، ويعني هذا كله أن العملية الأولى أخفقت والثانية نجحت لكن قد تم التراجع عنها، ويمكنك الذهاب الى الجدول لترى ان الزبون "aa" لم يتم حذفه.

ولفهم ماحدث ، فإننا قمنا بإضافة مستخدم موجود بالفعل ، وهذا ما يمنعه SQL Server إذ أنه من غير المسموح إضافة نفس القيمة للجدول ، وهو ما نوهنا عنه في سرد مزايا SQL Server ، الجملة الثانية الخاصة بحذف الزبون "aa" قد تم تنفيذها بالفعل ، لكن نظراً لأن ناتج الخطأ @maxerr لا يساوي صفر كما رأيت ، فتم التراجع عن عملية الحذف.

مثال آخر على اخفاق العملية الثانية:-

في هذا المثال سنقوم بإضافة زبون جديد ، ثم نقوم بحذف زبون ذو سجل فرعي child record في الجدول Orders ، قم بإضافة الزبون "aaa" ثم قم بحذف الزبون "ALFKI" عن طريق الاستعلام التالي:

exec sp_Trans_Test 'aaa', 'aaa' , 'ALFKI'

ليظهر لك الناتج كما بالشكل التالي:

```
SQLQuery2.sql - (...001\Hossam (57))*   SQLQuery1.sql - (...-001\Hossam (54))*
print 'INSERT error number:' + cast(@inserr as nvarchar(8))
print 'DELETE error number:' + cast(@delerr as nvarchar(8))
return @maxerr

exec sp_Trans_Test 'a ', 'a ', 'z '
Select * from Customers
exec sp_Trans_Test 'aa ', 'aa ', 'z '
exec sp_Trans_Test 'a', 'a', 'aa'
exec sp_Trans_Test 'aaa', 'aaa' , 'ALFKI'

Messages

(1 row(s) affected)
Msg 547, Level 16, State 0, Procedure sp_Trans_Test, Line 19
The DELETE statement conflicted with the REFERENCE constraint "FK_Orders_Customers". The conflict occurred
The statement has been terminated.
Transaction rolled back
INSERT error number:0
DELETE error number:547
```

ظهر ناتج الخطأ للجملة DELETE بالرقم 547 ، وتم التراجع عن العملية بنفس الكيفية التي تراجعت بها في المثال السابق ، على الرغم من نجاح العملية الأولي ، برجوع القيمة 0 للأخطاء ، إلا أنه تم التراجع عن العملية ككل بسبب إخفاق جملة الحذف لأن الزبون"ALFKI" يحوي سجل فرعي في الجدول Orders وتم مقارنة رقم الخطأ الناجم الذي لايساوي صفر فتم التراجع عن العملية كما رأيت ، وتستطيع التاكد من ذلك بالجملة

Select * from Customers

لترى ان الزبون "aaa" غير موجود.

مثال لإخفاق كلا العمليتين:-

جرب إدخال زبون موجود بالفعل ، وجرب حذف عنصر لايُمكن حذفه ،قم بكتابة الاستعلام التالي:

exec sp_Trans_Test 'a', 'a', 'ALFKI'

لترى النتيجة كما بالشكل:

```
SQLQuery2.sql - (...001\Hossam (57))*   SQLQuery1.sql - (...-001\Hossam (54))*
print 'INSERT error number:' + cast(@inserr as nvarchar(8))
print 'DELETE error number:' + cast(@delerr as nvarchar(8))
return @maxerr

exec sp_Trans_Test 'a ', 'a ', 'z '
Select * from Customers
exec sp_Trans_Test 'aa ', 'aa ', 'z '
exec sp_Trans_Test 'a', 'a', 'aa'
exec sp_Trans_Test 'aaa', 'aaa' , 'ALFKI'
exec sp_Trans_Test 'a', 'a', 'ALFKI'

Messages
Msg 2627, Level 14, State 1, Procedure sp_Trans_Test, Line 12
Violation of PRIMARY KEY constraint 'PK_Customers'. Cannot insert duplicate key in object 'dbo.Customers'.
The statement has been terminated.
Msg 547, Level 16, State 0, Procedure sp_Trans_Test, Line 19
The DELETE statement conflicted with the REFERENCE constraint "FK_Orders_Customers". The conflict occurred :
The statement has been terminated.
Transaction rolled back
INSERT error number:2627
DELETE error number:547
```

كما ظهر فى نافذة النواتج ، أنه تم التراجع عن العملية كلها ، بسبب اخفاق كلا الجملتين INSERT و DELETE بظهور أرقام الأخطاء 2627 للإضافة و 547 للحذف ، وتلاحظ أن الرسالة هذه المرة هي:

The statement has been terminated.

كتابة أكواد للعمليات باستخدام ADO.NET

في هذا المثال سنقوم بعمل برنامج مقابل للإجراء المخزن الذي كتبناه في الأمثلة السابقة sp_Trans_Test باستخدام الـ C#.

- قم بإنشاء مشروع جديد في فيجول ستوديو من النوع Windows Application وسمه TransTest
- قم بتسمية الملف Form1.cs الى Transaction.cs.
- قم بتغيير اسم الفورمة الى ADO.NET Transaction In C#.
- قم بإضافة 3 text boxes وكذا 3 labels وزر واحد ، لتصبح الفورمة كما بالشكل:



- قم بإضافة using directive للملف Transaction.cs:

using System.Data.SqlClient;

- قم بإدخال الكود التالي في الحدث button1_click للزر button1 بالضغط مرتين على الزر ليفتح لك الكود:

```
SqlConnection conn = new SqlConnection(@"data source = .;integrated security = true;
database = Northwind.mdf
");
// INSERT statement
string sqlins = @"
insert into customers(customerid,companyname)
values(@newcustid, @newconame) ";
// DELETE statement
string sqldel = @"
delete from customers
where customerid = @oldcustid";
// open connection
conn.Open();
SqlTransaction sqltrans = conn.BeginTransaction();
try
{
// create insert command
SqlCommand cmdins = conn.CreateCommand();
cmdins.CommandText = sqlins;
cmdins.Transaction = sqltrans;
cmdins.Parameters.Add("@newcustid",System.Data.SqlDbType.NVarChar, 5);
cmdins.Parameters.Add("@newconame",
System.Data.SqlDbType.NVarChar, 30);
// create delete command
SqlCommand cmddel = conn.CreateCommand();
cmddel.CommandText = sqldel;
cmddel.Transaction = sqltrans;
cmddel.Parameters.Add("@oldcustid",System.Data.SqlDbType.NVarChar, 5);
// add customer
cmdins.Parameters["@newcustid"].Value = textBox1.Text;
cmdins.Parameters["@newconame"].Value = textBox2.Text;
cmdins.ExecuteNonQuery();
// delete customer
cmddel.Parameters["@oldcustid"].Value = textBox3.Text;
cmddel.ExecuteNonQuery();
// commit transaction
sqltrans.Commit();

// no exception, transaction committed, give message
MessageBox.Show("Transaction committed");
}
catch (System.Data.SqlClient.SqlException ex)
{
// roll back transaction
sqltrans.Rollback();
MessageBox.Show(
"Transaction rolled back\n" + ex.Message,"Rollback Transaction");
}
catch (System.Exception ex)
{
MessageBox.Show("System Error\n" + ex.Message, "Error");
}
finally
{
// close connection
conn.Close();
}
```

- قم بتشغيل البرنامج عن طريق الضغط على F5 ثم ادخل القيم التالية على الترتيب : "b","bb", "bbb" كما كنا نفعل فى SQL Server بدلاً من القيم ""aaa ,"aa" ,"a" ثم اضغط على الزر Execute لكل قيمة على حدا ، واذا أردت أن تحذف أحدهم قم بادخاله في المربع الثالث لتحذفه ، لتظهر لك رسالة مفادها نجاح العملية.

لتوضيح ماحدث ، فإننا قمنا بفتح اتصال مع قاعدة البيانات ، ثم أنشأنا transaction ، مع ملاحظة أننا لا يمكننا انشاء اكثر من transaction مع الاتصال الواحد قبل التراجع عنه أو نجاحه ، ولن يعمل الـ transaction قبل فتح الاتصال conn.Open(); ، بعد ذلك قمنا بإدخال اوامر الاضافة والحذف ثم الحقناها بنفس الـ transaction :

```
// create insert command
SqlCommand cmdins = conn.CreateCommand();
cmdins.CommandText = sqlins;
cmdins.Transaction = sqltrans;
cmdins.Parameters.Add("@newcustid", SqlDbType.NVarChar, 5);
cmdins.Parameters.Add("@newconame", SqlDbType.NVarChar, 30);
// create delete command
SqlCommand cmddel = conn.CreateCommand();
cmddel.CommandText = sqldel;
cmddel.Transaction = sqltrans;
cmddel.Parameters.Add("@oldcustid", SqlDbType.NVarChar, 5);
```

ثم قمنا باسناد القيم المدخلة فى التكست بوكس إلى ما يناظرها من معاملات داخل الاستعلام :

```
// add customer
cmdins.Parameters["@newcustid"].Value = textBox1.Text;
cmdins.Parameters["@newconame"].Value = textBox2.Text;
cmdins.ExecuteNonQuery();
// delete customer
cmddel.Parameters["@oldcustid"].Value = textBox3.Text;
cmddel.ExecuteNonQuery();
```

ثم استعملنا الدالة sqltrans.Commit(); لبيان نجاح العملية او في حالة الاخفاق لنقوم بعمل تراجع استعملنا:

```
catch (System.Data.SqlClient.SqlException ex)
{
//Roll back transaction
sqltrans.Rollback();
}
```

خاتمة الفصل

في هذا الفصل قمنا بتغطية أساسيات الـ Transactions وماهية خصائصة المختلفة مثل ACID ، تعرفنا على العمليات الموزعة والمحلية ، كيفية كتابة كود العمليات باستخدام T-SQL ومن ثم تعرفنا على كيفية عمل ذلك عن طريق ADO.NET واستخدام C# 2010 ، لديك الآن الأساسيات للإنطلاق فى كتابة العمليات الخاصة بك وقتما تحتاج ، والمزيد من المهارات ستتعلمها فى الفصول القادمة إن شاء الله.

انتظروا الإصدار الثاني من الكتاب مكتملاً بباقي الفصول إن شاء الله